Logical foundations of games with imperfect information : uniform strategies

Bastien Maubert

ANNÉE 2014

UNIVERSITÉ DE RENNES 1

ueb

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1
Mention : Informatique

École doctorale Matisse

présentée par

Bastien MAUBERT

préparée à l'unité de recherche IRISA – UMR6074
Institut de Recherche en Informatique et Systèmes Aléatoires ISTIC
UFR Informatique et Électronique

Fondations logiques des jeux à information imparfaite : stratégies uniformes.

Thèse soutenue à Rennes le 17 Janvier 2014

devant le jury composé de :

Joseph Y. HALPERN
Professeur, Cornell University / Président

Christof LÖDING
Professeur associé, RWTH Aachen / Rapporteur

Ramaswamy RAMANUJAM
Professeur, IMSc Chennai / Rapporteur

Dietmar BERWANGER
Chargé de recherche, CNRS-ENS Cachan / Examinateur

Thomas BOLANDER
Professeur associé, DTU Lyngby / Examinateur

Cătălin DIMA
Professeur, Université Paris-Est Créteil / Examinateur

Sophie PINCHINAT
Professeur, Université Rennes 1 / Directrice de thèse

Guillaume AUCHER
Chaire INRIA-Université Rennes 1 / Co-directeur de thèse

He who turns himself into a beast gets rid of the pain of being a man.
Samuel Johnson

Remerciements

First of all I would like to thank all those who honoured me by participating in my jury: many thanks to Christof Loeding and Ramanujam for accepting the task of reviewing my work, and to Dietmar Berwanger, Thomas Bolander, Cătălin Dima and Joe Halpern for taking the time to read it and examine my defence. Joe, I am especially grateful that you managed to include this hook in Britany in your tight schedule, and that you did us the great favour of presiding the defence.

Bien évidemment je remercie du fond du coeur Sophie Pinchinat, sans qui je n'en serais clairement pas là. J'ose croire que je ferais quelque chose d'intéressant aussi, mais ça n'aurait peut-être rien à voir avec l'informatique. Ce qui est sûr c'est que ce que je fais actuellement me plaît énormément, et c'est en grande partie grâce à l'enthousiasme, à l'énergie et au dévouement que Sophie démontre chaque jour pour la recherche que j'y ai pris goût et que j'ai pu finir cette thèse. Bien entendu Guillaume Aucher a aussi joué un grand rôle durant ces trois années. Outre les fenêtres qu'il ma ouvertes vers le monde des philosophes, sa passion, sa rigueur scientifique et son perfectionnisme m'ont beaucoup appris. Je voudrais aussi remercier très chaleureusement François Schwarzentruber. Enseigner avec lui fut une expérience exceptionnelle. François est un génie de l'enseignement, même s'il jugera toujours que ses cours sont pourris. Et surtout, merci pour cet optimisme implacable, cette bonne humeur inébranlable, et cette touche de folie inénarrable.

Mais il n'y a pas que LogicA dans la vie, et je tiens à remercier aussi toutes les personnes que j'ai cotoyées pendant ces années à l'IRISA, et qui font que c'est un endroit merveilleux pour travailler : que ce soit les gens de Sumo, de Celtique, les assistants, que j'ai beaucoup embêtés mais qui ont toujours été parfaits, les filles de la cafète, toujours le sourire aux lèvres et le mot pour rire, et tous les autres que j'ai certainement oubliés. J'espère retrouver un endroit où travailler est si agréable.

Je remercie aussi infiniment toute ma famille, et spécialement mes parents, qui m'ont toujours encouragé dans toutes mes entreprises (excepté du point de vue vestimentaire et capillaire peut-être). Même si vous ne pouviez pas être là pour la soutenance on pensait beaucoup à vous, avec Daddy, Mémé et Tata Zabeth, qui m'ont fait un immense plaisir en venant y assister, et que je remercie encore une fois. Et Mamie, je sais que tu aurais aimé être là, alors tu as le droit à un merci spécial aussi :-) Et un gros bisou au frangin et à la frangine ! Merci d'être venus à la fête post-soutenance, ça comptait beaucoup pour moi que vous soyiez là.

Bien sûr je dis un grand merci à tous les potes, Renaud, Clémence, Bartek, Fab, Roumy, Loic, Chloé, Marine, Toupet, Damien, Marie, Nils, Vincent, Sim, Marco, La Piechte, Léa, Harmonie, Pépé, Beber, AD, FX, Jean, Mesnard, Fanny, Raton, Maël, Thibaut, Jerem, Coco, François, Loulou, Pépette, Victor, Victor, Bret, Papa, Mouton, Anne, Pamela, Laura, Reda, ceux que je vois tous les jours comme ceux que je ne vois que trop rarement, ceux qui sont venus à la soutenance comme ceux qui n'ont pas pu (pour diverses raisons...). Il va sans dire que si toutes ces années à Rennes ont été aussi magiques c'est grâce à vous tous, et à plein d'autres encore (m'en voulez pas).

Et pour terminer, merci Ketsia, pour tout.

Contents

Résumé long en français

0.1 Contexte

La théorie des jeux est un domaine de recherche riche, vaste et dynamique, dont l'objet est l'étude mathématique rigoureuse de la prise de décisions stratégiques. L'appellation standard "théorie des jeux" recouvre en fait une grande variété de formalismes différents. Cependant, les ingrédients principaux d'une théorie des jeux sont toujours un ensemble de *joueurs*, une description de l'*information* et des *actions* disponibles pour chaque joueur à chaque point de décision, les *gains* pour chaque joueur dans chaque résultat possible, et un *concept de solution* qui décrit ce qu'est une solution désirable. Ces concepts de solutions font toujours référence à la notion de *stratégie*, c'est à dire à des fonctions assignant aux joueurs, pour chaque situation possible, quel coup jouer. Les caractéristiques et les hypothèses faites sur ces différents ingrédients varient selon l'objet étudié. Quand le domaine a émergé durant la première moitié du XXème siècle avec les travaux de Von Neumann et Morgenstern, les problèmes étudiés étaient surtout d'ordre économique. Par conséquent, les premiers travaux considérèrent des jeux avec certaines caractéristiques, et en particulier des jeux à *durée finie*. Dans de tels jeux, les joueurs prennent des décisions (simultanément ou à leur tour) un nombre fini de fois, après quoi un résultat est atteint. Cependant le paradigme des jeux, au fil du siècle dernier, a suscité l'intérêt de nombreuses sciences telles que la philosophie, les sciences politiques, la biologie, les mathématiques fondamentales et, plus récemment, la logique et l'informatique. La diversité des objectifs et des systèmes étudiés a donné lieu à une taxonomie complexe des jeux : forme normale/extensive, somme nulle/non-nulle, information parfaite/imparfaite, information complète/incomplète, tour par tour/concurrent, durée finie/infinie...

Dans cette thèse nous considérons les jeux joués sur des graphes (où un noeud est appelé *position* et une arête un *coup*), et plus précisément nous nous intéressons aux propriétés qui caractérisent une "bonne" stratégie. La propriété la plus évidente est qu'une bonne stratégie devrait être *gagnante* (*i.e.* elle assure le joueur qui la suit de toujours gagner) pour une condition de gain donnée, dont la forme dépend du type de jeu considéré. Mais dans certains cas, des contraintes supplémentaires sont imposées pour restreindre l'ensemble des stratégies disponibles pour un joueur. Nous décrivons certaines de ces propriétés qui ont émergé et/ou pris beaucoup d'importance en informatique théorique et en logique pour l'informatique. Tout d'abord, nous rappelons les motivations principales pour considérer en informatique des jeux à *durée infinie*.

Jeux à durée infinie

Une tâche qui a reçu une attention considérable au long des dernières décennies est la vérification des systèmes critiques. En effet, alors qu'on assigne à des systèmes automatiques la responsabilité d'effectuer des tâches de plus en plus complexes, et dans des domaines – tels que l'avionique ou les centrales électriques – où des fautes peuvent avoir des conséquences tragiques, le besoin de développer des méthodes pour garantir la "correction" de tels systèmes est indéniable. Une des approches les plus employées et le *model checking*. Cela consiste à représenter les systèmes à vérifier par des abstractions mathématiques adaptées (appelées *modèles*), exprimer les propriétés à vérifier dans un certain langage logique, et développer des algorithmes qui vérifient automatiquement qu'une formule donnée est vraie sur un modèle donné. Selon le type de propriété considéré, on utilise différentes logiques et différents types de modèles. Prenons l'exemple d'une centrale électrique. Une propriété simple qui est attendue est : "Dans tous les comportements possibles, la centrale n'explose jamais". Pour exprimer de telles propriétés temporelles des systèmes/programmes, les logiques temporelles ont été étudiées intensivement. Les logiques temporelles classiques sont LTL, introduite par Pnueli (1977) et qui traite de propriétés du *temps linéaire*, CTL, étudiée initialement par Clarke and Emerson (1981) et qui considère des propriétés du *temps arborescent*, et CTL*, qui fut introduite par Emerson and Halpern (1983) et combine le pouvoir d'expressivité de LTL et CTL. Des algorithmes efficaces pour le model checking de ces logiques ont été développés, et ont mené au développement d'outils maintenant largement utilisés dans l'industrie – voir *e.g.* Vardi (2008) pour un bref survol historique. Une approche en quelque sorte duale au model checking, la *synthèse de programme*, s'appuie aussi sur ces logiques temporelles : au lieu de vérifier qu'un programme donné vérifie certaine propriété, le but est, à partir d'une spécification logique, de synthétiser automatiquement (le squelette d') un programme qui, par construction, vérifie cette spécification. Un travail fondateur dans cette approche, pour des spécifications CTL, est dû à Clarke and Emerson (1981).

A noter que toutes ces logiques temporelles – LTL, CTL et CTL* – sont des fragments du μ-calcul modal de Kozen (1983). Le μ-calcul est d'une grande importance, car il représente en un sens le pendant logique des automates alternants sur les objets infinis (Muller and Schupp, 1987), qui sont des machines très puissantes qui fournissent des procédures de décision pour de nombreuses logiques temporelles. A l'inverse des automates classiques sur les mots finis, les automates travaillant sur des objets infinis ont des exécutions infinies, et nécessitent des conditions d'acceptation adaptées. Les plus importantes sont les conditions d'acceptation de Büchi, Rabin, Street, Muller et parité. Le μ-calcul est plus étroitement lié à la condition de parité, car c'est elle qui vient le plus naturellement lorsqu'on traduit des formules du μ-calcul vers les automates d'arbres alternants. A leur tour, ces automates de parité sont intimement liés aux jeux à durée infinie. En particulier, les jeux à durée infinie et condition de gain de parité fournissent une procédure de décision naturelle pour tester la vacuité du langage des automates d'abres de parité, et donc la satisfiabilité des formules de logiques temporelles. Ces connections donnent lieu à une théorie puissante des automates, logiques et jeux à durée infinie (Grädel et al., 2002).

Les jeux à durée infinie sont aussi un outil très naturel pour représenter et raisonner à propos de systèmes qui *interagissent*. De tels systèmes sont communs, et peuvent être aussi simples qu'un protocole de communication entre une imprimante et ses utilisateurs.

Pour revenir à l'exemple de la centrale électrique, considérons un système (un *contrôleur*) chargé de contrôler la centrale. Une propriété souhaitable est : "Quoiqu'il se passe dans la centrale, le contrôleur peut toujours réagir pour assurer que la centrale n'explose pas". Cette propriété met en jeu une *alternance* entre deux entités actives : à chaque fois que quelque chose se passe dans la centrale, le contrôleur doit pouvoir prendre des mesures appropriées, auxquelles la centrale peut réagir, et ce indéfiniment. Ce genre de propriété, bien qu'exprimable dans le μ-calcul modal, peut être capturé de manière bien plus intuitive par la notion de stratégie gagnante dans un jeu à durée infinie. Dans l'exemple, nous voulons assurer que le contrôleur a une stratégie gagnante contre la centrale, où la condition de gain pour le contrôleur est d'empêcher que la centrale n'explose.

Au final, les jeux à durée infinie sont un outil intuitif et puissant pour non seulement tester la vacuité des automates de parité, et donc résoudre les problèmes de satisfiabilité des logiques temporelles, mais aussi exprimer des propriétés complexes de systèmes, mettant en jeu une alternation entre différentes entités. Ces deux aspects font des jeux à durée infinie des objets centraux de l'informatique théorique moderne. Cependant, afin de capturer bon nombre de situations du monde réel, il manque aux modèles que nous avons considéré pour l'instant l'aspect *information imparfaite*.

Jeux à information imparfaite

Dans la vie de tous les jours il est courant de prendre des décisions sans avoir en main toutes les informations pertinentes – pensons par exemple au jeu de Poker. En informatique, cette situation peut se rencontrer quand certaines variables d'un système sont internes/privées. Dans notre exemple de centrale électrique, le contrôleur peut n'avoir accès qu'à la température interne du réacteur mais pas à la pression, à cause d'un équipement endommagé. En théorie des jeux, ceci est modélisé en rassemblant dans des *ensembles d'informations* les situations qu'un joueur ne peut pas différencier. Durant une partie, le joueur ne connaît pas forcément la situation courante, et ne peut donc pas baser sa stratégie dessus. Cela mène à imposer qu'une stratégie pour ce joueur doit assigner le même coup à jouer dans chaque situation d'un même ensemble d'informations. Cette contrainte a des conséquences profondes sur l'existence de stratégies gagnantes et le problème de décider cette existence. Par exemple, tandis que les jeux ω-réguliers à deux joueurs et information parfaite sont déterminés, *i.e.* il y a toujours une stratégie gagnante pour l'un des deux joueurs, ce n'est plus le cas lorsqu'on ajoute l'aspect information imparfaite. De plus, alors que par essence les conditions ω-régulières sont évaluées sur des parties individuelles, indépendamment des autres parties résultant d'une stratégie, l'information imparfaite engendre la nécessité de considérer des *ensembles* de parties, afin de vérifier qu'une stratégie est consistente sur les ensembles d'informations. La solution classique est de rassembler toutes les parties indistingables en procédant à une construction par sous-ensembles, réduisant ainsi le problème à la résolution d'un jeu à information parfaite équivalent (Reif, 1984). Plus récemment, un algorithme basé sur les antichaines a été développé (Chatterjee et al., 2006; Berwanger et al., 2010), qui évite cette coûteuse construction par sous-ensembles en utilisant une représentation succinte des ensembles d'informations.

Dans les jeux à information imparfaite, l'information disponible pour un joueur est souvent représentée par une *capacité observationnelle* : le joueur ne voit pas les positions directement, mais en perçoit une abstraction, appellée *observations*, et différentes positions

peuvent partager la même observation. Pour cette raison, certains travaux (Chatterjee et al., 2006) utilisent le terme de *stratégie basée sur les observations* pour désigner les stratégies soumises à la contrainte de jouer le même coup dans des situations indistingables. D'autres travaux (Berwanger and Doyen, 2008; Berwanger et al., 2010) définissent les stratégies directement sur les observations au lieu de les définir sur les positions. Parce que ces stratégies doivent être définies "uniformément" sur les ensembles d'informations, elles sont quelquefois appelées *stratégies uniformes* dans la communauté des logiques stratégiques (van Benthem, 2001, 2005; Jamroga and van der Hoek, 2004). Nous emploierons parfois le terme *stratégies à information imparfaite* pour désigner ce type de stratégies.

Dans les jeux extensifs, où les stratégies sont définies sur les séquences de positions, décrire comment chaque position est observée par le joueur n'est, en général, pas suffisant pour caractériser les ensembles d'informations. Il est aussi nécessaire de spécifier quelles sont les *capacités mémorielles* du joueur. Cela amène à distinguer entre *mémoire parfaite* et *mémoire imparfaite*. Dans le premier cas, le joueur se souvient de l'intégralité des observations qu'il a reçues au cours d'une partie, alors que dans le second cas il oublie une partie de l'information. Un joueur à mémoire imparfaite peut-être *sans mémoire*, *i.e.* il ne se souvient de rien et prend ses décisions uniquement en fonction de l'observation de la position courante. Il peut aussi avoir une *mémoire bornée*, ce qui signifie qu'il peut stocker une certaine quantité finie d'information ; ou il peut avoir une *mémoire non bornée*, mais oublier quand même des parcelles d'information.

Tandis que les jeux à information imparfaite et mémoire parfaite ont fait l'objet de nombreux travaux (Reif, 1984; Chatterjee et al., 2006; Berwanger and Doyen, 2008; Berwanger et al., 2010), le cas de la mémoire imparfaite a reçu beaucoup moins d'attention depuis que des paradoxes concernant leur interprétation ont été soulevés (Piccione and Rubinstein, 1997). Néanmoins, la mémoire imparfaite permet de modéliser des problèmes pertinents : typiquement, certaines unités informatisées n'ont qu'un espace mémoire très limité et ne peuvent se souvenir de parties arbitrairement longues. De plus, un article récent plaide en faveur d'une étude systématique des jeux avec mémoire imparfaite (Berwanger et al., 2012).

Il est important de remarquer que les hypothèses faites sur la mémoire d'un joueur, l'existence de stratégies gagnantes à information imparfaite et la difficulté de décider cette existence sont étroitement liées. La mémoire donne du pouvoir : en effet, un joueur sans mémoire peut perdre un jeu dans lequel un joueur à mémoire parfaite posséderait une stratégie gagnante, grâce à l'information supplémentaire dont il dispose. Cependant, donner plus de mémoire aux joueurs peut aussi augmenter la complexité de résoudre les jeux. Par exemple, les jeux à information imparfaite avec trois joueurs à mémoire parfaite sont indécidables (Peterson et al., 2001; Berwanger and Kaiser, 2010), alors que les mêmes jeux sont trivialement décidables pour des joueurs sans mémoire.

Jeux sémantiques pour les logiques à information imparfaite

Nous avons vu que dans les jeux à information imparfaite, les stratégies sont soumises à une certaine contrainte d' "uniformité" : elles doivent jouer le même coup dans les différentes situations d'un même ensemble d'informations. Les *logiques à information imparfaite*, qui ajoutent des notions de dépendence et/ou indépendance à la logique du premier ordre, fournissent un second exemple de contrainte sur les stratégies admissibles. Rappelons que

la sémantique des formules de la logique du premier ordre peut être donnée par un jeu entre deux joueurs, le Vérificateur et le Contradicteur. Pour la logique d'indépendence de Hintikka and Sandu (1989), on obtient une sémantique de jeu naturellement en ajoutant de l'information imparfaite dans la sémantique de jeux de la logique du premier ordre, et en utilisant la notion habituelle de stratégies à information imparfaite. Cependant, cela ne fonctionne plus pour la Logique de Dépendence, introduite plus récemment par Väänänen (2007). A la place, Väänänen définit une contrainte ad hoc sur les stratégies autorisées pour le Vérificateur, et il appelle cette contrainte une "contrainte d'uniformité". Cette contrainte ressemble à celle des stratégies à information imparfaite et rend aussi ces jeux non-déterminés, raison pour laquelle Väänänen a d'abord qualifié les jeux obtenus de jeux à information imparfaite. Cependant il n'est pas clair que ces jeux puissent être définis en termes de jeux à information imparfaite au sens classique du terme.

Ce problème est bien connu dans la communauté Logique de Dépendence. Récemment, un certain nombre d'autres logiques à information imparfaite ont été étudiées (Grädel and Väänänen, 2013; Galliani, 2012; Engström, 2012). Toutes ajoutent à la logique du premier ordre de nouveaux atomes qui capturent des notions de la théorie des dépendences dans les bases de données. Elles ont aussi en commun que leur sémantique peut être définie naturellement en ajoutant, dans le jeu sémantique de la logique du premier ordre, des contraintes sur les stratégies du Vérificateur. Comme pour la Logique de Dépendence, ces jeux ne sont pas à proprement parler des jeux à information imparfaite, et afin d'étudier les propriétés de ces jeux sémantiques, Grädel (2013) a récemment introduit une notion de *jeux d'atteignabilité du second ordre*. Ces jeux généralisent les jeux sémantiques des logiques de l'information imparfaite, et ils sont qualifiés de *second ordre* car, comme pour les stratégies à information imparfaite, les contraintes sur les stratégies concernent des *ensembles* de parties résultant des stratégies.

Conditions de gain épistémiques

Les conditions de gain impliquant des aspects épistémiques fournissent un troisième exemple de propriétés de stratégies qui concernent des ensembles de parties, et qui sont aussi intrinsèquement liées à l'information imparfaite. De telles conditions de gain (ou *objectifs*) sont prédominants dans le domaine florissant des systèmes multi-agents. Reprenons l'exemple de la centrale électrique. Supposons que la stratégie du contrôleur n'était pas gagnante. Maintenant la centrale a explosé, et une mission doit être envoyée pour chercher des survivants, sécuriser le lieu et le nettoyer. Parce que la centrale était nucléaire, il est impossible d'envoyer des humains, aussi une équipe de robots est-elle déployée. Pour accomplir leur mission, les robots doivent interagir et prendre des décisions sous information imparfaite. En effet, ils ne connaissent pas la configuration exacte du site, et leurs capteurs peuvent être endommagés par les radiations. Mais en plus de ces aspects que nous avons déjà évoqués, ils doivent être capables de raisonner à propos de leurs connaissances, ainsi que celles des autres robots et d'éventuels survivants. Par exemple, un robot doit pouvoir être capable de raisonner en ces termes : "Je crois que cet humain est peut être en vie, donc je vais le secourir", ou "Je sais que cet humain est mort, donc je continue à chercher des survivants". De plus, leurs objectifs eux-mêmes peuvent inclure des aspects épistémiques. Par exemple : "Je dois mettre au point une stratégie qui m'assure que le robot médical, au bout d'un certain temps, obtiendra la connaissance du fait que cet humain est en vie".

Les systèmes logiques traitant de l'incertitude sont légion. Les premiers travaux s'intéressaient à la représentation et le raisonnement à propos de la connaissance dans un contexte statique (Sato, 1977; Lehmann, 1984; Fagin et al., 1991). Ces logiques épistémiques ont ensuite été étendues, dans le contexte des systèmes distribués, pour étudier les intéractions entre connaissance et temps ; les logiques obtenues forment la famille des Logiques Temporelles Épistemiques (ETL) (Parikh and Ramanujam, 1985; Ladner and Reif, 1986; Halpern and Vardi, 1989; Fagin et al., 1995; van der Meyden and Shilov, 1999; Dima, 2008). Une autre approche pour introduire du dynamisme dans la logique épistémique est la Logique Épistémique Dynamique (DEL), qui peut être vue comme une adaptation de la logique propositionnelle dynamique (PDL) au contexte épistémique, et où les actions sont des évènements informatifs (Baltag et al., 1998; van Ditmarsch et al., 2007; van Benthem, 2011). Le problème de synthétiser des protocoles avec des objectifs épistémiques et temporels a aussi reçu une certaine attention (van der Meyden and Vardi, 1998; van der Meyden and Wilke, 2005). Plus récemment, les Logiques Épistémiques Temporelles ont été combinées avec les Logiques du Temps Alternant (Alur et al., 2002) dans le but de raisonner à propos de la connaissance, du temps et d'aspects stratégiques dans un langage logique unifié ; les logiques résultantes sont appelées Logiques Épistémiques du Temps Alternant (van der Hoek and Wooldridge, 2003; Jamroga and van der Hoek, 2004; Dima et al., 2010). Ces logiques très expressives contiennent donc, en plus d'opérateurs temporels et d'opérateurs épistémiques, des opérateurs qui permettent de quantifier sur des stratégies avec des conditions de gain impliquant temps, connaissance et capacités stratégiques.

Clairement, les stratégies avec des conditions de gain épistémiques sont un autre exemple de stratégies qui doivent prendre en compte des ensembles de parties, quoique pas de la même manière que les stratégies à information imparfaite ou les stratégies pour les logiques à information imparfaite. Notons que dans toutes ces logiques, comme pour les jeux à information imparfaite classiques, les hypothèses faites sur les capacités observationnelles des agents sont cruciales. Plusieurs propriétés qui caractérisent les capacités des agents sont habituellement considérées : mémoire parfaite, sans mémoire, synchrone, asynchrone, pas de miracles... Différentes combinaisons de ces hypothèses donnent des résultats différents pour les logiques correspondantes, et ces résultats sont en général prouvés au cas par cas.

Point sur la situation et problème de recherche

Comme nous l'avons décrit, il semble que dans un nombre croissant de domaines, des propriétés de stratégies impliquant des ensembles de parties jouent un rôle centrale, la plupart du temps à cause de l'information imparfaite. Cependant nous rappelons que pour le cas de la Logique de Dépendence et autres logiques à information imparfaite, il n'est pas clair que les contraintes imposées sur les stratégies puissent être définies en termes d'information imparfaite.

Le tableau global est complexe, surtout parce que le temps et la connaissance sont essentiellement orthogonaux. Pour comprendre dans quelle mesure la connaissance et le temps sont effectivement orthogonaux, il est utile de voir les stratégies come des arbres infinis : le temps, lié à l'ordonnancement des positions rencontrées dans les parties (les branches) et aux branchements (les choix), concerne la dimension *verticale* des arbres. La connaissance, en revanche, relie différentes parties qui partagent la même information, et

concerne donc la dimension *horizontale*.

A notre connaissance, il n'existe pas d'étude approfondie de contraintes sur les stratégies impliquant cette dimension horizontale. Dans cette thèse nous proposons d'étudier en profondeur une notion générale de stratégies soumises à des contraintes impliquant des ensembles de parties. Le but est de clarifier le tableau, de mieux comprendre la complexité de résoudre les jeux avec des contraintes horizontales sur les stratégies, et de développer des techniques génériques qui dépendent le moins possible d'hypothèses particulières faites sur les capacités observationnelles et mémorielles des agents.

0.2 Contribution

Parce que notre intérêt principal n'est pas la concurrence mais plutôt les propriétés des stratégies, nous choisissons de travailler dans le cadre simple des jeux à deux joueurs joués en tour par tour sur des graphes. Néanmoins, toute la théorie que nous développons peut être adaptée à des modèles plus sophistiqués, tels que les structures de jeux concurrents (de Alfaro et al., 1998; de Alfaro and Henzinger, 2000). Nous commençons par rappeler dans le Chapitre 2 un certain nombre de définitions et de résultats connus sur les classes de complexité, les jeux à durée infinie, la logique propositionnelle et du premier ordre, les automates sur des objets infinis, et certaines classes de relations sur les mots représentables de manière finie.

Dans le Chapitre 3 nous définissons une notion générale de *stratégies uniformes*. Les *propriétés d'uniformité* des stratégies sont exprimées dans un langage logique, $\mathcal{L}_{\rightsquigarrow}$, qui est essentiellement CTL^* augmentée de deux quantificateurs sur les parties *reliées*, $\boxminus$ et $\boxbox$. Les motivations pour choisir ce langage sont les suivantes.

Premièrement, $\mathcal{L}_{\rightsquigarrow}$ est proche des logiques temporelles épistémiques classiques (l'opérateur de connaissance K de ces logiques est un cas particulier de nos quantificateurs $\boxminus$ et $\boxbox$), ce qui nous permet d'exprimer les propriétés d'uniformité de manière intuitive. Nous illustrons cet aspect en montrant que les trois types de propriétés horizontales discutées plus haut se reformulent aisément sous forme de stratégies uniformes. Deuxièmement, bien que $\mathcal{L}_{\rightsquigarrow}$ ne contienne pas le μ-calcul, cette logique combine l'expressivité des logiques du temps linéaire et du temps arborescent. De plus, nous contenter de CTL^* comme base pour la dimension verticale de nos propriétés d'uniformité nous permet de nous concentrer sur la dimension horizontale, qui est l'objet principal de notre étude. Troisièmement, utiliser une logique temporelle classique nous permet de bénéficier des techniques automates connues pour aborder le problème de synthétiser des stratégies uniformes.

A propos de l'introduction de nos deux quantificateurs $\boxminus$ et $\boxbox$, il existe dans la littérature deux approches différentes pour donner la sémantique de la connaissance quand des stratégies sont en jeu. Dans van der Meyden and Vardi (1998) et van der Meyden and Wilke (2005) par exemple, le but est de synthétiser des stratégies qui vérifient quelque objectif temporel et épistémique. Dans ces travaux, la sémantique de l'opérateur de connaissance suggère que lorsqu'une stratégie est fixée, tous les agents *connaissent la stratégie*, et par conséquent ils ne considèrent plus possibles des séquences d'évènements qui ne suivent pas cette stratégie. D'un autre côté, dans les Logiques Épistémiques du Temps Alternant, la sémantique de la connaissance ne dépend pas des stratégies prises, ce qui implique que les agents *ne connaissent pas les stratégies*. Il semble que les hypothèses menant à

ces différentes sémantiques de la connaissance ne sont habituellement pas explicitées. Par l'introduction des deux quantificateurs $\boxminus$ et $\boxplus$, nous tentons de clarifier cette idée. Intuitivement, le quantificateur *strict* $\boxminus$ correspond au premier cas, où les agents connaissent la stratégie, tandis que le quantificateur *plein* $\boxplus$ modélise des agents qui ignorent la stratégie, comme dans le second cas. Les noms "strict" et "plein" reflètent le fait que la portée du quantificateur strict $\boxminus$ est restreinte aux parties suivant la stratégie, tandis que le quantificateur plein $\boxplus$ porte sur la totalité des parties possibles dans le jeu. Ces deux quantificateurs, et notre volonté d'étudier des propriétés des stratégies très générales, font que la sémantique de $\mathcal{L}_{\rightsquigarrow}$ est quelque peu originale à deux égards.

La première originalité concerne la sémantique du quantificateur plein, $\boxplus$. Comme pour CTL^*, une formule de $\mathcal{L}_{\rightsquigarrow}$ est interprétée sur un arbre. De par l'utilisation que nous faisons de nos formules, à savoir exprimer des propritétés d'uniformité de stratégies, ces arbres sont amenés à représenter des stratégies dans un jeu. Cependant, pour donner la sémantique du quantificateur plein $\boxplus$, il faut aussi représenter l'ensemble des parties possibles dans un jeu. Pour cela chaque modèle comporte, en plus d'un arbre, une forêt (ensemble d'arbres) appelé l'*univers*.

La seconde originalité concerne les relations donnant la sémantique de nos quantificateurs $\boxminus$ et $\boxplus$. Usuellement dans ETL, la sémantique de l'opérateur de connaissance K est une quantification universelle sur les histoires reliées à l'histoire courante par quelque relation binaire. Les modèles les plus généraux autorisent des relations arbitraires entre les histoires, et nous suivons cette approche. Cependant, tous les travaux dont nous avons connaissance qui considèrent des questions algorithmiques où les modèles font partie de l'entrée du problème (essentiellement model-checking), se restreignent à des relations *générées finiment*. Dans la plupart des cas, une relation d'équivalence est donnée sur les positions de l'arène (ou les états du modèle). Cette relation représente les observations d'un agent, et elle est étendue sur les histoires selon certaines hypothèses faites sur les capacités de l'agent – mémoire parfaite/imparfaite, synchrone/asynchrone (Halpern and Vardi, 1989). Au lieu de cela, lorsque nous étudions le problème de synthétiser des stratégies uniformes, nous considérons des relations *reconnues finiment*. Plus précisément, nous considérons la classe des *relations rationnelles*, qui sont les relations reconnaissables par des automates à deux bandes, aussi appelés *transducteurs* (Eilenberg, 1974; Berstel, 1979). Il est intéressant de noter que la plupart des relations considérées en logique épistémique temporelle, et notamment celles générées finiment, évoquées plus haut, sont reconnues par des tranducteurs assez simples. De plus, les relations rationnelles ne sont pas forcément des relations d'équivalence, ce qui permet de capturer, par exemple, des relations utilisées en révision des connaissances et pour modéliser la plausibilité, avec des axiomatisations K45 ou KD45 (Fagin et al., 1995).

Après avoir donné la syntaxe et la sémantique de $\mathcal{L}_{\rightsquigarrow}$, nous définissons notre notion de stratégies uniformes, puis nous l'illustrons en revisitant plusieurs notions de la littérature. Dans certains cas, comme pour les stratégies à information imparfaite ou les stratégies pour les logiques à information imparfaite, le quantificateur strict est nécessaire. Cependant nous montrons que capturer des stratégies avec des conditions de gain épistémiques requiert parfois l'emploi du quantificateur plein. Nous abordons ensuite notre problème principal, qui est la synthèse de stratégies uniformes. Pour ce faire, nous considérons d'abord séparément les deux types de quantificateurs, le strict et le plein, car leurs propriétés diffèrent

et mènent à employer des techniques différentes. Cette séparation donne les Chapitres 4 et 5 respectivement.

Nous commençons dans le Chapitre 4 avec les propriétés d'uniformité exprimées dans $\mathcal{L}_{\sim}$ mais n'utilisant que le quantificateur strict. Les stratégies soumises à ce type de propriétés sont appelées *stratégies strictement uniformes*. Le problème que nous considérons consiste à, étant données une arène de jeu, un transducteur reconnaissant une relation binaire entre les parties, et une formule de $\mathcal{L}_{\sim}$ sans quantificateur plein représentant une propriété d'uniformité, décider s'il existe une stratégie uniforme pour le joueur 1. Nous prouvons que, sans surprise, l'existence de stratégie strictement uniforme est indécidable pour la classe des relations rationnelles. Plus précisément, nous prouvons en encodant le problème de l'existence de stratégie distribuée dans les jeux de sûreté à information imparfaite à trois joueurs (Peterson et al., 2001; Berwanger and Kaiser, 2010), que l'existence de stratégie strictement uniforme est indécidable même pour les relations d'*équivalence* reconnues par des transducteurs *synchrones*. Dans le but de mieux comprendre la complexité de ce problème, nous introduisons la notion d'*automates d'arbres bondissants*. Ces automates sont équipés d'une relation binaire sur les mots, et ils étendent les automates d'arbres alternants classiques en autorisant des sauts entre des noeuds reliés dans l'arbre d'entrée. Le problème de l'existence de stratégie strictement uniforme se réduit au problème de la vacuité des automates d'arbres bondissants, ce qui implique l'indécidabilité de ce dernier problème pour des automates équipés de relations rationnelles.

Néanmoins, nous établissons que si les automates d'arbres bondissants sont équipés de *relations reconnaissables*, une sous classe de relations rationnelles qui, informellement, ne considèrent qu'une information bornée sur chacune des branches, alors ces automates peuvent être simulés par des automates d'arbres à deux sens classiques (Vardi, 1998). De ce résultat nous obtenons une procédure de décision pour l'existence de stratégies strictement uniformes pour la classe des relations reconnaissables. De plus, nous prouvons que ce problème est 2-Exptime-complet, c'est à dire qu'il n'est essentiellement pas plus complexe que tester la satisfiabilité de CTL^* (Emerson, 1990), ou que résoudre des jeux avec de simples conditions de gain LTL (Pnueli and Rosner, 1989). Enfin, grâce aux techniques classiques pour extraire d'un automate d'arbre une représentation finie d'un modèle, notre procédure de décision permet de synthétiser une stratégie uniforme lorsqu'il en existe une.

Dans le Chapitre 5 nous tournons notre attention vers le problème de la synthèse de *stratégies pleinement uniformes*, c'est à dire de stratégies dont les propriétés d'uniformité sont spécifiées dans $\mathcal{L}_{\sim}$ mais en n'utilisant que le quantificateur plein. Nous prouvons que le problème est décidable mais nonélémentaire pour la classe complète des relations rationnelles. Plus précisément, nous établissons que le problème est k-Exptime-complet pour les propriétés d'uniformité avec une profondeur d'imbrication de quantificateurs $\boxtimes$ d'au plus k (2-Exptime-complet si $k \leq 2$). Notre procédure de décision repose sur l'introduction d'*automates d'ensembles d'informations*, que nous utilisons comme outils pour calculer des ensembles d'informations définis par des relations rationnelles. Les automates d'ensembles d'informations encapsulent une partie de la difficulté technique de notre procédure. Ils nous permettent aussi d'identifier une sous-classe de relations rationnelles (K45NM) pour laquelle la complexité du problème de l'existence de stratégies pleinement uniformes s'écroule de nonélémentaire à 2-Exptime-complet. Ce résultat est intéressant car, premièrement, K45NM contient encore la plupart des relations habituellement considérées en logique

épistémique temporelle et jeux à information imparfaite, et la complexité du problème pour cette classe de relations est, comme dans le cas décidable des stratégies strictement uniformes, la même que celle de résoudre les jeux LTL.

Après avoir établi ces résultats pour les stratégies strictement uniformes et pleinement uniformes dans le cas d'une relation, nous étendons notre langage dans le Chapitre 6 pour considérer plusieurs relations $\rightsquigarrow_i$ et leurs quantificateurs associés, $\boxminus_i$ et $\boxplus_i$. Nous appelons $n\mathcal{L}_{\rightsquigarrow}$ le langage étendu pour n relations. Nous montrons d'abord que tous nos résultats de décidabilité et de complexité établis dans le cas d'une relation tiennent toujours pour plusieurs relations, sauf dans un cas. En effet, décider l'existence de stratégies pleinement uniformes pour la classe de relations K45NM n'est plus 2-Exptime-complet mais devient nonélémentaire dès que deux relations sont autorisées. Nous établissons des bornes de complexité précises, à savoir que le problème est h-Exptime-complet pour des propriétés d'uniformité de profondeur d'*alternation* entre différents quantificateurs d'au plus h (2-Exptime-complet si $h \leq 2$). Nous identifions aussi un fragment de $n\mathcal{L}_{\rightsquigarrow}$ qui permet d'utiliser des quantificateurs stricts *et* pleins dans la même formule, et pour lequel l'existence de stratégie uniforme est encore décidable, et a essentiellement la même complexité que s'il n'y avait que des quantificateurs pleins. Ce cadre généralisé à plusieurs relations capture aisément plusieurs problèmes de la litérature. En particulier, nous montrons que nos résultats fournissent une preuve unifiée de plusiseurs résultats connus sur le model-checking des logiques temporelles épistémiques, pour différentes capacités des agents ; en fait, le model checking de ces logiques est décidable pour toutes relations d'indistingabilité rationnelles. Nous décrivons aussi comment la résolution de jeux à plusieurs joueurs, information imparfaite et objectifs temporels et épistémiques peut être réduite à un problème de stratégie uniforme dans un jeu à deux joueurs, avec une propriété d'uniformité impliquant un quantificateur strict pour chaque joueur du jeu original.

Une autre application de nos résultats concerne le sujet florissant de la synthèse de protocoles épistémiques qui, grossièrement, vise à synthétiser des protocoles (ou stratégies) dans des situations où l'information et la connaissance jouent un rôle crucial dans le comportement du système et/ou dans les objectifs des agents impliqués. Dans le Chapitre 7 nous considérons d'abord le problème de *planification épistémique* (Bolander and Andersen, 2011; Löwe et al., 2011; Aucher and Bolander, 2013), qui est un cas particulier de problème de planification dans le cadre de la Logique Épistémique Dynamique. Etant donnés une représentation de la situation épistémique initiale, un ensemble d'évènements épistémiques possibles et un objectif sous forme de formule épistémique, le problème de planification épistémique consiste à synthétiser une séquence finie d'évènements qui, lorsqu'on l'exécute dans la situation initiale, mène à une situation épistémique qui satisfait l'objectif. Ce problème est indécidable pour plusieurs agents, mais restreindre le type d'évènements autorisés donne un problème qui a récemment été prouvé décidable (Yu et al., 2013). Nous réduisons cette variante à un problème de stratégie uniforme, et obtenons une preuve alternative de décidabilité ainsi que des bornes supérieures de complexité. De plus, notre approche nous permet de synthétiser sans coût supplémentaire un automate fini qui engendre tous les plans solution du problème. Pour terminer, nous considérons une notion de protocole épistémique et son problème de synthèse associé, qui généralise la planification épistémique selon plusieurs directions. Premièrement, les plans ne sont plus des séquences d'évènements finies mais infinies. Deuxièmement, nous cherchons un protocole,

i.e. un arbre d'évènements (ou un ensemble de plans), au lien d'un seul plan. Troisièmement, l'objectif n'est plus de type accessibilité, mais peut être une quelconque formule temporelle épistémique (avec des opérateurs de connaissance stricts et pleins). Une fois de plus, nos résultats sur les stratégies uniformes nous permettent de résoudre ce problème général et de synthétiser des protocoles épistémiques.

Notes

Une première version de notre notion de stratégies uniformes et une étude de son expressivité ont été publiées dans Maubert and Pinchinat (2012). Une partie des résultats présentés dans les Chapitres 3 et 5 a aussi été publiée dans Maubert and Pinchinat (2014) et Maubert et al. (2013). Ce dernier travail a été mené en collaboration avec Laura Bozzelli, à qui est dûe la preuve de bornes inférieures décrite en Appendice A. A noter que dans ces travaux, notre langage de spécification des propriétés d'uniformité était basé sur LTL au lieu de CTL* comme c'est le cas ici. Nous avons choisi de passer de LTL à CTL* car cela accroît clairement l'expressivité, comme l'atteste la capacité de capturer *e.g.* le module checking de Kupferman and Vardi (1997), sans affecter nos résultats.

Dans le Chapitre 3, nous illustrons l'utilisation du quantificateur plein avec l'exemple des *jeux à condition d'opacité*. Ce sont des jeux avec une condition de gain épistémique particulière qui ont fait l'objet d'une étude antérieure, qui a mené à deux publications : nous avons introduit ces jeux dans Maubert and Pinchinat (2009) et étudié leur complexité dans Maubert et al. (2011).

Les résultats du Chapitre 4 ont aussi été publiés dans Maubert and Pinchinat (2013). La plupart des résultats des Chapitres 3, 4, 5 et 6 ont été acceptés pour publication dans une édition spéciale d'Information and Computation pour le 1er Workshop International sur le Raisonnement Stratégique (SR 2013), qui s'est tenu à Rome en Mars 2013.

Finalement, nous présentons dans le Chapitre 7 quelques applications de nos méthodes dans le cadre de la Logique Épistémique Dynamique. Durant cette thèse, nous avons aussi mené d'autres travaux dans ce domaine, en collaboration avec Guillaume Aucher et François Schwarzentruber.

Nous décrivons informellement l'idée de ces travaux. DEL permet d'exprimer des propriétés :

1. d'une situation épistémique initiale,
2. d'un évènement informatif qui se produit dans cette situation, et
3. de la situation résultant de l'occurence de cet évènement.

Le problème des *séquents DEL* est le suivant. Étant données trois formules (une pour chaque point), est-il vrai que, si une situation épistémique initiale vérifie la première formule et qu'un évènement vérifie la seconde, alors la situation résultante vérifiera la troisième ? Nous avons prouvé que ce problème est décidable en produisant une méthode tableau pour la validité des séquents DEL. Nous avons aussi établi que le problème est NExptime-complet. Ces résultats ont été publiés dans Aucher et al. (2011). Nous avons ensuite généralisé cette méthode tableau au cas d'une séquence de n évènements, ainsi qu'au cas de relations d'accessibilités sérielles ou réflexives dans les modèles. Ces résultats ont été publiés dans Aucher et al. (2012).

Chapter 1

Introduction

1.1 Context

Game theory is a rich, vast and dynamic field of research, which aims at the rigorous mathematical study of strategic decision making. Though the standard designation "game theory" actually covers a wide range of different formalisms, the main ingredients of any game theory are always a set of *players*, a description of the *information* and *actions* available to each player at each decision point, the *payoffs* for each player in each possible outcome, and a *solution concept* describing what is a desirable solution. Solution concepts always refer to *strategies*, *i.e.* functions that prescribe to a player what move should be made in each possible situation. The characteristics and assumptions made on these ingredients vary depending on the object of study. When the field emerged during the first half of the 20th century, with the work of Von Neumann and Morgenstern, it was mostly directed towards economic concerns. Therefore, studies initially focused on games with particular features, one of them being *finite duration*. In such games, players make decisions (either simultaneously or in turn) a finite number of times, after which an outcome is reached. Along the past century however, the game theoretic paradigm raised interest in a number of fields such as philosophy, political science, biology, pure mathematics and, more recently, logics and computer science. The variety of purposes and systems under study lead to a whole intricate taxonomy of games – normal/extensive form, zero/nonzero sum, perfect/imperfect information, complete/incomplete information, turn-based/concurrent, finite/infinite duration...

In this thesis we are interested in games played on graphs (where a node is called a *position* and an edge is a *move*), and especially in the properties that characterize "good" strategies. The most obvious property is that a strategy should be *winning* (*i.e.* it ensures that the player who follows it always win) for some winning condition, the form of which depends on the type of game considered. But in some settings, additional constraints are imposed to limit the set of allowed strategies that a player can use. We describe some of these properties of strategies that have emerged and/or gained much importance in theoretical computer science and logics for computer science. First, we recall one of the main motivations for considering games of *infinite duration* in computer science.

1.1.1 Infinite duration games

One task that has received a tremendous amount of attention over the past decades is that of verifying critical systems. Indeed, as automatic systems are given the responsibility to perform more and more complex tasks, and in domains – such as avionics or power plants – where the consequences of faults can be tragic, the need to develop methods to ensure the "correctness" of such systems cannot be questioned. One of the most successful approaches to this end is *model checking*. In this approach, systems to be verified are represented by suitable mathematical abstractions (called *models*), properties to be ensured are expressed in some logical language, and algorithms are developed to automatically check that a given model verifies a given formula. According to the type of properties considered, different logics and models are used. Let us take the example of a power plant. One simple property that should be verified by the plant is: "In all possible behaviours, the power plant never explodes". To express such properties of systems/programs involving temporal aspects, temporal logics have been intensively studied. Classic temporal logics are LTL, that was introduced by Pnueli (1977) and deals with properties of *linear time*, CTL, first studied by Clarke and Emerson (1981) and concerned with *branching time* properties, and CTL*, also called the *full branching time logic*, that was introduced by Emerson and Halpern (1983) and combines the expressive power of both LTL and CTL. Efficient algorithms for the model checking of these logics have been developed, and have led to the development of tools now widely used in the industry – see *e.g.* Vardi (2008) for a brief historical overview. Based on these temporal logics, an approach somehow dual to model checking is called *program synthesis*: instead of checking that a given program verifies some property, the aim is, given a logical specification, to automatically synthesize a (skeleton of a) program that verifies this specification by construction. A foundational work in this trend, for CTL specifications, is due to Clarke and Emerson (1981).

Note that all these temporal logics – LTL, CTL and CTL* – are fragments of the modal μ-calculus of Kozen (1983). μ-calculus is of great importance, as it is in a sense the logical counterpart of alternating automata on infinite objects (Muller and Schupp, 1987), which are very powerful machines that provide decision procedures for many logics of programs. Note that unlike classic automata on finite words, automata working on infinite objects have infinite runs, and they need adapted acceptance conditions. The most important ones are Büchi, Rabin, Street, Muller and parity acceptance conditions. μ-calculus is more closely related to the parity condition, as translating μ-calculus formulas into alternating tree automata naturally leads to using the parity condition. In turn, these parity alternating tree automata are deeply related to games of infinite duration. In particular, infinite-duration games with parity winning condition provide a natural decision procedure for testing the language nonemptiness of parity tree automata, and thus the satisfiability of temporal logics formulas. These connections lead to a powerful theory of automata, logics and infinite games (Grädel et al., 2002).

Infinite duration games also provide a very natural way to model and reason about systems that *interact*. Such systems are very common, and can be as simple as a communication protocol between a printer and its users. Back to the example of the power plant, consider a system in charge of controlling the plant (a *controller*). One desirable property is the following: "Whatever happens in the plant, the controller can always take actions to ensure that the plant never explodes". This property involves *alternation* between two en-

tities taking actions: every time something happens in the plant, the controller must have the possibility to take appropriate measures, to which the plant may react, so long and so forth. This kind of property, though expressible in the modal μ-calculus, can be captured much more intuitively in terms of winning strategies in an infinite duration game. In the example, we want to ensure that the controller has a winning strategy against the plant, where the winning condition for the controller is to prevent the plant from exploding.

In the end, infinite duration games are an intuitive and powerful tool not only for testing the emptiness of parity tree automata, and thus solving satisfiability problems for temporal logics, but also for expressing complex properties of systems that involve alternation between several entities. These two aspects make infinite duration games central in modern computer science. However, in order to capture many real-world situations, the models we have discussed so far lack the *imperfect information* aspect.

1.1.2 Games with imperfect information

In everyday life it is common to make decisions without having all the relevant information in hand – consider for instance Poker games. In computer science this situation occurs for example when some variables of a system are internal/private. In our power plant example of Section 1.1.1, the controller may only have access to the temperature inside the reactor but not to the pressure, because of some damaged equipment. In game theory, this is modeled by gathering situations that a player cannot distinguish in *information sets*. During a play, the player cannot tell which situation in the current information set is the actual one, and therefore she cannot base her strategy on the exact current situation. This leads to requiring that a strategy must assign the same move to every situation inside an information set. This constraint has deep impacts on the existence of winning strategies, and the problem of deciding this existence. For example, while ω-regular two-player games with perfect information are determined, *i.e.* there is always a winning strategy for one of the players, this is no longer the case for games with imperfect information. Also, while in essence ω-regular conditions are evaluated on individual plays, independently of other plays that result from a strategy, turning to imperfect-information games raises the need to deal with *sets* of plays to check that a strategy is consistent over information sets. The classic solution is to gather all indistinguishable plays by performing a powerset construction, thus reducing the problem to solving an equivalent perfect-information game (Reif, 1984). More recently, an algorithm based on antichains has been developed (Chatterjee et al., 2006; Berwanger et al., 2010), that avoids the costly powerset construction by using succinct representations of information sets.

In extensive games with imperfect information, the information available to a player is often represented by some *observational ability*: the player does not see positions, but only has access to an abstraction of these, so-called *observations*, and several positions can share the same observation. For this reason, some works (Chatterjee et al., 2006) use the term *observation-based strategies* to refer to strategies submitted to the constraint of playing the same move in indistinguishable situations. Other works (Berwanger and Doyen, 2008; Berwanger et al., 2010) define strategies directly on observations instead of positions. Because these strategies must be defined "uniformly" over information sets, this kind of strategies is also referred to as *uniform strategies* in the community of strategic logics (van Benthem, 2001, 2005; Jamroga and van der Hoek, 2004). We may also sometimes refer to

these strategies as *imperfect-information strategies*.

In extensive games, where strategies are defined on sequences of positions, describing how each position is observed by a player is, in general, not sufficient to characterize information sets. One needs in addition to specify what the player's *memory abilities* are. This yields the distinction between *perfect recall* and *imperfect recall*. In the former, the player remembers the whole history of the observation she had of a play so far, while in the latter the player forgets part of the information. A player with imperfect recall can be *memoryless*, *i.e.* she does not remember anything and takes decisions only according to the observation of the current position. She can also have *bounded memory*, which means that she can remember a certain finite amount of information; or she can have an unbounded memory, but she does not perfectly remember the history and loses pieces of information.

While games with imperfect information and perfect recall have been studied intensively (Reif, 1984; Chatterjee et al., 2006; Berwanger and Doyen, 2008; Berwanger et al., 2010), the case of imperfect recall has received much less attention since paradoxes concerning the interpretation of such games were raised (Piccione and Rubinstein, 1997). Nonetheless, relevant problems may be modeled with imperfect recall: typically, particular computing resources have very limited memory and cannot remember arbitrarily long histories. Furthermore, a recent work has advocated the relevance of a systematic study of games with imperfect recall (Berwanger et al., 2012).

It is important to notice that the assumptions made on the memory of the player, the existence of observation-based winning strategies, and the difficulty of deciding this existence are tightly related. Memory gives power to the players: indeed, a memoryless player may lose a game where a player with perfect recall, having more information, may possess a winning strategy. But giving more memory to the players may also increase the complexity of solving games. For example, solving imperfect-information games with three players and perfect recall is undecidable (Peterson et al., 2001; Berwanger and Kaiser, 2010), while the problem is trivially decidable for memoryless players.

1.1.3 Semantic games for logics of imperfect information

We have seen that in games with imperfect information, strategies are submitted to some "uniformity" constraint: they must assign the same move to the different situations of a same information set. A second example of a constraint on admissible strategies, which is related to the first one, comes from *logics of imperfect information*. These logics add notions of dependence and/or independence in first-order logic, and their semantics can be given by two-player games with special constraints on the strategies. Recall that the semantics of first-order logic formulas can be given by a two-player game between a Verifier and a Falsifier. For the Independence Friendly logic of Hintikka and Sandu (1989), one obtains a natural game semantics by adding imperfect information in the semantic games of first-order logic, using the classic notion of imperfect-information strategies. However, this does not work for the more recent Dependence Logic, introduced by Väänänen (2007). Instead, Väänänen defines an ad hoc constraint on the strategies allowed for the Verifier, and he calls this constraint a "uniformity requirement". This constraint is reminiscent of imperfect-information strategies and makes these games undetermined, which is why Väänänen originally qualified these semantic games as imperfect-information games. However, it is not clear whether these games can be defined in terms of imperfect-information

games in the usual acceptation of the notion.

This issue is well-known in the community of Dependence Logic. Recently, a number of other logics of imperfect information have been studied (Grädel and Väänänen, 2013; Galliani, 2012; Engström, 2012), that add in first-order logic new atoms capturing notions from database dependence theory. They all have in common that their semantics are naturally given by adding some constraints on strategies in the semantic games of first-order logic. Like for Dependence Logic, these games are not exactly games with imperfect information, and in order to study their properties, Grädel (2013) recently introduced a notion of *second-order reachability games*. These games generalize the semantic games for logics of imperfect information, and they are called *second order* because, like for imperfect-information strategies, the constraints on strategies concern sets of plays in the outcome.

1.1.4 Epistemic winning conditions

Yet another property of strategies that concerns sets of plays, and is also intrinsically based on imperfect information, concerns winning conditions involving epistemic aspects. Such objectives are predominant in the thriving topic of multi-agent systems. We illustrate this by returning to our power plant example. Assume that the strategy of the controller was not winning. Now the plant has exploded, and a mission must be carried out to look for survivors, secure the place and clean it. Because it was a nuclear power plant, we cannot send humans and a team of robots is deployed. To achieve their mission, the robots must interact and take decisions under imperfect information. Indeed, they do not know the exact configuration of the site, and their sensors may be damaged by radiations. But in addition to these features that we already discussed, they must be able to reason about their knowledge as well as the other robots and possible survivors' one. For example, a robot should be able to reason in the following terms: "I believe that this human may still be alive, therefore I rescue him", or "I know for sure that this human is dead, therefore I continue searching for survivors". Moreover, their objectives themselves may include epistemic aspects. For example: "I need to devise a strategy to make sure that the medical robot eventually knows that this human is alive".

Logical systems concerned with uncertainty are many. The first works on the matter were concerned with representing and reasoning about knowledge in a static setting (Sato, 1977; Lehmann, 1984; Fagin et al., 1991). These epistemic logics have then been extended, in the context of distributed systems, to study interactions between knowledge and time; logics of this kind form the family of Epistemic Temporal Logics (ETL) (Parikh and Ramanujam, 1985; Ladner and Reif, 1986; Halpern and Vardi, 1989; Fagin et al., 1995; van der Meyden and Shilov, 1999; Dima, 2008). Another approach to add dynamics in epistemic logic is Dynamic Epistemic Logic, which can be seen as an adaptation of propositional dynamic logics to the epistemic setting, and where actions are informative events (Baltag et al., 1998; van Ditmarsch et al., 2007; van Benthem, 2011). The problem of synthesizing protocols with epistemic temporal objectives has also received some attention (van der Meyden and Vardi, 1998; van der Meyden and Wilke, 2005). More recently, Epistemic Temporal Logics have been combined with Alternating-time Temporal Logic (Alur et al., 2002) in order to reason about knowledge, time and strategies in a unified language. These very expressive logics are called Alternating-time Temporal Epistemic

Logics (van der Hoek and Wooldridge, 2003; Jamroga and van der Hoek, 2004; Dima et al., 2010). In addition to temporal and epistemic operators, they contain operators that quantify over strategies verifying some property involving time, knowledge and strategic abilities.

Obviously, strategies with epistemic objectives are another instance of strategies that must deal with sets of plays, yet not in the same manner as imperfect-information strategies or strategies for logics of imperfect information. In all these logics, like for classic games with imperfect information, the assumptions made on the observational power of agents are crucial. A number of properties are usually considered to characterize different capabilities of agents: perfect recall, memoryless, synchronous, asynchronous, no miracles... Different combinations of these assumptions yield different results for the corresponding logics, and results are usually proven on a case-by-case basis.

1.1.5 Review of the situation and research problem

As shown above, there seems to be a growing number of domains where properties of strategies involving sets of plays are central, most of the time because of imperfect information. However, recall that in the case of Dependence Logic and other logics of imperfect information, it is not clear how the constraint imposed on strategies can be defined in terms of imperfect information.

The whole picture is intricate, mainly because time and knowledge are essentially orthogonal, as we show below. This orthogonality yields a complex theoretical universe to reason about. In order to understand to which extent knowledge and time are indeed orthogonal, viewing strategies as infinite trees is helpful: time is about the *vertical* dimension of the trees as it relates to the ordering of encountered positions along plays (branches) and to the branching in the tree. On the contrary, knowledge is about the *horizontal* dimension, as it relates plays carrying the same information.

To the best of our knowledge, there is no thorough study of general constraints on strategies involving this horizontal dimension. In this thesis we propose to study in depth a general notion of strategies with constraints involving sets of plays. The aim is to clarify the picture, to better understand the complexity of solving games with horizontal constraints on strategies, and to develop generic techniques that depend as little as possible on particular assumptions made on observational and memory abilities of agents.

1.2 Contribution and structure of the document

Even though studying strategic abilities of multiple players is a central concern in game theory and multi-agent systems, we choose to initiate this study of general horizontal constraints of strategies in the simple framework of turn-based games played on graphs between two players, in which we will study the existence of "good" strategies for one of these players. However, the notions we define may be adapted to more sophisticated models such as concurrent game structures (de Alfaro et al., 1998; de Alfaro and Henzinger, 2000), and in addition we will see that some strategic problems concerning multiple players can be expressed already in our two-player setting (see Section 6.5.2).

We first recall in Chapter 2 various definitions and known results concerning complexity classes, infinite games, propositional and first order logics, automata on infinite objects

and classes of finitely representable relations on words.

In Chapter 3 we define a general notion of *uniform strategies*. The *uniformity properties* of strategies are expressed in a logical language, $\mathcal{L}_{\rightsquigarrow}$, which is CTL^* augmented with two quantifiers over *related* plays, $\boxminus$ and $\boxplus$. The motivations for choosing this language are the following.

First, $\mathcal{L}_{\rightsquigarrow}$ is close to classic Epistemic Temporal Logics (the knowledge operator K of these logics is a particular case of our quantifiers $\boxminus$ and $\boxplus$), which allows us to express uniformity properties in an intuitive manner. This is illustrated in Chapter 3, where we show that the three kinds of horizontal properties discussed in Sections 1.1.2, 1.1.3 and 1.1.4 are easily rephrased as uniform strategies. Second, even though $\mathcal{L}_{\rightsquigarrow}$ does not contain the full μ-calculus, it still combines the full expressiveness of linear-time and branching-time logics. Settling for CTL^* as basis for the vertical dimension of our uniformity properties allows us to focus on the horizontal dimension, which is the major object of our study. Third, using a classic temporal logic, we can benefit from well-known automata techniques to tackle the problem of synthesizing uniform strategies.

Concerning the introduction of our two quantifiers, $\boxminus$ and $\boxplus$, note that there are, in the literature, two distinct approaches to give the semantics of knowledge when strategies are involved. In van der Meyden and Vardi (1998) and van der Meyden and Wilke (2005) for example, the aim is to synthesize strategies that verify some epistemic temporal objective. In these works, the semantics of the knowledge operator suggest that when a strategy is fixed, all the agents *know the strategy*, and therefore they no longer consider possible sequences of events that do not follow this strategy. On the other hand, in Alternating-time Temporal Epistemic Logics, the semantics of the knowledge do not depend on the strategies taken, implying that the agents *do not know the strategies*. It seems that the assumptions leading to these definitions of knowledge are usually not made explicit. The introduction of the two quantifiers $\boxminus$ and $\boxplus$ is an attempt to clarify this idea. Intuitively, the *strict* quantifier $\boxminus$ corresponds to the first case, where agents know the strategy, while the *full* quantifier $\boxplus$ models agents who ignore the strategy, as in the second case. The names "strict" and "full" reflect the fact that the range of the strict quantifier $\boxminus$ is *restricted* to the outcomes of a strategy, while the full quantifier $\boxplus$ ranges over the *full* set of possible plays in the game. Because of these two quantifiers and our will to study very general properties of strategies, the semantics of $\mathcal{L}_{\rightsquigarrow}$ is somewhat original, in two regards.

The first originality concerns the semantics of the full quantifier, $\boxplus$. As for CTL^*, $\mathcal{L}_{\rightsquigarrow}$ formulas are interpreted on trees. Because we use formulas to specify properties of strategies, these trees are meant to represent strategies in some game. However, giving the semantics of the full quantifier $\boxplus$ requires to represent the set of all possible plays in the game. To this aim, each model contains, in addition to a tree, a forest (set of trees) called the *universe*.

The second originality concerns the binary relations that give the semantics of our quantifiers $\boxminus$ and $\boxplus$. Classically in ETL, the semantics of the knowledge operator K is a universal quantification over histories related to the actual one. The most general frameworks allow for arbitrary relations between histories, and we follow this approach. However, in all the works that we know of that consider algorithmic questions where the models are part of the problem's input (mostly model checking), only *finitely generated* relations are considered. In most cases, an equivalence relation on the positions of the

arena (or the states of the model) is given. This relation represents the observations of some agent, and it is extended to histories according to certain hypothesis made on the agent's abilities – perfect/imperfect recall, synchronous/asynchronous (Halpern and Vardi, 1989). In contrast, when we study the problem of synthesizing uniform strategies, we consider *finitely recognized* relations. More precisely, we consider the class of *rational relations*, which are relations recognized by two-tape automata also called *finite state transducers* (Eilenberg, 1974; Berstel, 1979). Noticeably, most equivalence relations used in epistemic temporal logics, and in particular the finitely generated ones above described, are recognized by fairly simple transducers. Additionally, rational relations need not be equivalences, but they also encompass *e.g.* relations used in belief revision, and for modelling plausibility, with K45 or KD45 axiomatization (Fagin et al., 1995).

After giving the syntax and the semantics of $\mathcal{L}_{\rightsquigarrow}$, and after defining our notion of uniform strategies, we illustrate it by revisiting several notions from the literature. Some of these, like imperfect-information strategies and strategies for logics of imperfect information, require the use of the strict quantifier. We show that, however, capturing strategies with epistemic objectives sometimes necessitates the full quantifier instead. We then turn to the study of our main problem, which is the synthesis of uniform strategies. To do so, we first consider separately the two kinds of quantifiers, strict and full, as they have different properties and require different techniques. This yields Chapters 4 and 5 respectively.

We start in Chapter 4 with uniformity properties expressed in $\mathcal{L}_{\rightsquigarrow}$ but using the strict quantifier only. Strategies subject to this kind of properties are called *strictly-uniform strategies*. The *strictly-uniform strategy problem* consists in, given a game arena, a transducer accepting some binary relation between plays, and a uniformity property expressed in $\mathcal{L}_{\rightsquigarrow}$ without full quantifiers, deciding whether Player 1 has a uniform strategy. We establish that, unsurprisingly, the strictly-uniform strategy problem is undecidable for rational relations. More precisely, we prove by encoding the distributed strategy problem in safety games with imperfect information (Peterson et al., 2001; Berwanger and Kaiser, 2010), that the strictly-uniform strategy problem is undecidable even for *equivalence* relations recognized by *synchronous* transducers. In order to better understand the complexity of the problem, we introduce the notion of *jumping tree automata*. These automata are equipped with some binary relation over words, and they extend classic alternating tree automata by allowing for jumps in the input tree between related nodes. The strictly-uniform strategy problem reduces to the nonemptiness problem for jumping tree automata, which entails the undecidability of the latter problem for automata equipped with rational relations.

Nonetheless, we establish that if jumping tree automata are equipped with *recognizable relations*, a subclass of rational relations that basically only challenge a bounded amount of information in each branch, then they can be simulated by classic two-way tree automata (Vardi, 1998). From this result we derive a decision procedure for the strictly-uniform strategy problem with recognizable relations. Moreover, we prove the problem to be 2-ExpTime-complete, *i.e.* it has the same complexity as testing the satisfiability of CTL^* (Emerson, 1990), or solving games with simple LTL winning conditions (Pnueli and Rosner, 1989). Finally, classic techniques to extract a finitely represented model of a tree automaton allow us to synthesize a uniform strategy whenever one exists.

In Chapter 5 we turn our attention to the problem of synthesising *fully-uniform strategies*, *i.e.* strategies whose properties are specified in $\mathcal{L}_{\rightsquigarrow}$ but using the full quantifier only.

We prove that the problem is decidable but nonelementary for the whole class of rational relations. More precisely, we establish that the problem is k-EXPTIME-complete for uniformity properties that involve up to k nested $\boxbslash$ quantifiers – 2-EXPTIME-complete if $k \leq 2$. Our decision procedure relies on the introduction of *information set automata*, that we use as a tool to compute information sets defined by rational relations. Information set automata encapsulate a part of the technical difficulty of our procedure, and they also enable us to identify a subclass of rational relations (K45NM) for which the fully-uniform strategy problem collapses to 2-EXPTIME-complete. This result is of interest as, first, K45NM still contains most relations considered in Epistemic Temporal Logics and games with imperfect information, and the complexity of the problem for this class of relations is, as in the decidable case for strictly-uniform strategies, the same as solving LTL games.

After establishing these results for strictly-uniform and fully-uniform strategies in the case of one relation, in Chapter 6 we extend our language to allow for several relations $\rightsquigarrow_i$ and corresponding quantifiers $\boxminus_i$ and $\boxbslash_i$. We call $n\mathcal{L}_{\rightsquigarrow}$ the extended language for n relations. We show that all our results still hold when several relations are allowed, except for one case. Indeed, the fully-uniform strategy problem with K45NM relations is no longer 2-EXPTIME-complete, but it becomes nonelementary already for two relations. The precise complexity of the problem is h-EXPTIME-complete for uniformity properties whose depth of *alternation* between different quantifiers is at most h – 2-EXPTIME-complete if $h \leq 2$. We also identify a fragment of $n\mathcal{L}_{\rightsquigarrow}$ that allows for both strict and full quantifiers in the same formula and for which the uniform strategy problem is still decidable, and has the same complexity as with only full quantifiers. Several problems from the literature fit in this generalized setting. In particular, we show that several known results on model checking logics of knowledge and time, for different abilities of agents, find a unified proof in our work; in fact the model checking of such logics is decidable for all rational indistinguishability relations. We also describe how solving games with several players, imperfect information and epistemic temporal objectives can be reduced to solving a uniform strategy problem in a two-player game, where the uniformity property involves a strict quantifier for each player of the original game.

Another application of our results concerns the thriving topic of epistemic protocol synthesis, which basically aims at synthesizing protocols (or strategies) in situations where knowledge and information play a crucial role in the behaviour of the system and/or in the objectives of the agents involved. In Chapter 7 we first consider the *epistemic planning* problem (Bolander and Andersen, 2011; Löwe et al., 2011; Aucher and Bolander, 2013), which is an instance of planning problem in the framework of Dynamic Epistemic Logic (DEL). Given a representation of the initial epistemic situation, a set of possible epistemic events and an objective epistemic formula, the epistemic planning problem consists in synthesizing a finite sequence of events that, when triggered from the initial situation, yields a situation that verifies the objective epistemic formula. The problem for several agents is undecidable, however, a restriction on the allowed type of events yields a problem that has recently been proved decidable (Yu et al., 2013). We reduce this version of the problem to a uniform strategy problem, providing an alternative decidability proof as well as accurate upper bounds on the time complexity of the problem. Also, our approach allows us to synthesize at no cost a finite automaton that generates all the solution plans of the problem. We finally consider a notion of epistemic protocol and its associated

synthesis problem, which generalizes epistemic planning in several directions. First, plans are no longer finite but infinite series of events. Second, we look for a protocol, *i.e.* a tree of events (or a set of plans), instead of a single plan. Third, the objective is no longer reachability-like, but can be any temporal epistemic formula (with strict and full knowledge operators). Once again, our results on uniform strategies enable us to solve this general problem and synthesize epistemic protocols.

The contributions are discussed at the end of each chapter, along with related work and future research. We summarize these conclusions at the end of the document.

Notes

An early version of our notion of uniform strategies and a study of its expressivity have been published in Maubert and Pinchinat (2012). Also, part of the results presented in Chapters 3 and 5 have been published in Maubert and Pinchinat (2014) and Maubert et al. (2013). The latter work was done in collaboration with Laura Bozzelli, and the hardness proof described in Appendix A is due to her. Note that in these works our language to specify uniformity properties was based on LTL, instead of CTL* as is the case here. We decided to switch from LTL to CTL* because, while it clearly increases expressivity, as exemplified by the ability to capture *e.g.* module checking (Kupferman and Vardi, 1997), allowing for the full branching time logic leaves our results unchanged.

In Chapter 3, we illustrate the use of the full quantifier with the example of *games with opacity condition*. These are games with a particular epistemic winning condition that was the object of an earlier study, which led to two publications: we introduced these games in Maubert and Pinchinat (2009) and studied their complexity in Maubert et al. (2011).

Also, the results of Chapter 4 have been published in Maubert and Pinchinat (2013). Most of the results of Chapters 3, 4, 5 and 6 have been accepted for publication in a special edition of Information and Computation following the 1st International Workshop on Strategic Reasoning, that was held in Rome in March 2013.

Finally, we present in Chapter 7 some applications of our methods in the framework of Dynamic Epistemic Logic. More work on this topic has been carried out during this thesis, in collaboration with Guillaume Aucher and François Schwarzentruber.

We describe informally the idea of these works. Roughly speaking, DEL enables to express properties of:

1. an initial epistemic situation,
2. an informative event occurring in this situation, and
3. the resulting situation after the event has occurred.

The *DEL sequent* problem is the following. Given three formulas (one for each point), is it true that if an initial situation verifies the first formula and an event verifies the second, then the resulting situation will verify the third formula? We proved that this problem is decidable by providing a tableau method for the validity of DEL sequents. We also established that the problem is NExptime-complete. These results were published in Aucher et al. (2011). The tableau method was then generalized to the case of a series of n events, and also extended to the case where accessibility relations in the models are serial, and to the case where they are reflexive. These results were published in Aucher et al. (2012).

Chapter 2

Preliminaries

In this chapter we remind various classic notions and we fix some notations that we will use throughout this thesis. In the first section we pose some elementary definitions of complexity theory, and in the second one we fix some definitions for finite words, infinite words and infinite trees, which are ubiquitous in this thesis. The third section contains the definitions of game arenas, strategies and strategy trees. We also recall some classic results concerning infinite duration parity games. In the next section we briefly recall propositional and first order logics. The fifth section defines automata, both for words and trees, as we extensively use automata as tools to solve the problems addressed in this work. We also remind some central theorems concerning automata on infinite trees. The last section is about multi-tape word automata, also called finite state transducers, and their connections with various classes of relations that are at the heart of our results.

We start with some very basic notations concerning sets and relations. $\emptyset$ denotes the empty set; for two sets A and B, $|A|$ is the cardinal of A, 2^A is the powerset of A, *i.e.* the set of all subsets of A, $A \uplus B$ is the disjoint union of A and B, and $A \times B$ is the Cartesian product of A and B, *i.e.* the set of pairs (a, b) with $a \in A$ and $b \in B$; this generalizes to the product of n sets. A n-ary relation R between n sets $A_1, \ldots, A_n$ is a subset of $A_1 \times \ldots \times A_n$. For a binary relation $R \subseteq A \times B$, two elements $a \in A$ and $b \in B$ are said to be R-related if $(a, b) \in R$, which we may write $a\,R\,b$. When it is clear which relation is concerned, we may just say that a and b are related. For a relation $R \subseteq A \times B$ and an element $a \in A$, $R(a)$ denotes the set of elements in B related to a, *i.e.* $R(a) = \{b \in B \mid a\,R\,b\}$. Similarly, for an element b in B, $R^{-1}(b) = \{a \in A \mid a\,R\,b\}$. More generally, for a $n+1$-ary relation $R \subseteq A_1 \times \ldots \times A_{n+1}$ with $n \geq 1$, and a tuple $(a_1, \ldots, a_n) \in A_1 \times \ldots \times A_n$, we let $R(a_1, \ldots, a_n) = \{a \in A_{n+1} \mid (a_1, \ldots, a_n, a) \in R\}$. Finally, for two relations $R_1 \subseteq A \times B$ and $R_2 \subseteq B \times C$, the composition $R_1 \circ R_2 \subseteq A \times C$ of the relations is defined by $R_1 \circ R_2 = \{(a, c) \mid a \in A, c \in C, \exists b \in B \text{ such that } (a, b) \in R_1 \text{ and } (b, c) \in R_2\}$.

2.1 Main complexity classes

Computational complexity theory is about measuring, classifying and comparing the difficulty of computational problems. Roughly speaking, a problem is "hard" if the best algorithm to solve it requires a lot of resources for its execution. The two resources used by an algorithm that are most often considered are the time of computation and the memory

space required. Of course, for a given algorithm, these quantities may depend on the size of the input. In this work, we will usually define the size of a graph as its number of edges, the size of a logical formula as the number of symbols, and the size of an input as the sum of the sizes of its elements.

We now define some of the main space and time complexity classes. For convenience, we introduce iterated exponential functions as follows:

Definition 1. For all $k, n \in \mathbb{N}$, $\exp^0(n) = n$ and $\exp^{k+1}(n) = 2^{\exp^k(n)}$.

We will classically note PSPACE for the class of problems solvable in polynomial space, and for each $k \in \mathbb{N}$ we let k-EXPTIME be the class of problems that can be solved in time $\exp^k(n^c)$ for some constant $c \in \mathbb{N}$ (n is the size of the input).

We also define the class ELEMENTARY of *elementary* problems, *i.e.* problems that can be solved by an algorithm of worst-case time complexity a tower of exponentials of bounded height:

$$\text{ELEMENTARY} = \bigcup_{k \in \mathbb{N}} k\text{-EXPTIME}.$$

If a problem is decidable but not elementary it is *nonelementary*.

Note that in this work we will often define decision problems as the set of their positive instances, *i.e.* the set of instances for which the answer to the problem is *yes*.

2.2 Words and trees

In all this work, when we talk about an *alphabet* Σ we mean a finite set of *symbols*.

Finite and infinite words

For an alphabet Σ, Σ^* is the set of all *finite words* over Σ, ϵ denotes the *empty word*, $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$ is the set of nonempty finite words and Σ^ω is the set of infinite words. For a finite word w we note $|w|$ its *length*. For a nonempty finite word $w = a_1 \dots a_n$, $\text{last}(w) = a_n$ is its last symbol. For two finite words $w = a_1 \dots a_n$ and $w' = b_1 \dots b_m$, $w \cdot w' = a_1 \dots a_n b_1 \dots b_m$ is the *concatenation* of w and w'. A finite word w is a *prefix* of a word w', written $w \preccurlyeq w'$, if there exists a word w'' such that $w \cdot w'' = w'$. Finally, for a word $w = a_1 \dots a_n$, we note $\overline{w} = a_n \dots a_1$ its *mirror* word.

Infinite trees

A *tree alphabet* Υ is a finite set of *directions*. A tree is a subset of Υ^+ that is closed for nonempty prefixes, and such that all words in the tree start with the same direction, called the root. Note that in many works, the root of a tree is the empty word ϵ. However, when considering trees that represent strategies in a game, it is convenient for us to see their root as the initial position of the game. This justifies our slightly unusual definition. We also define forests which, intuitively, can be seen as unions of trees. Finally, because the games we define in the next section have only infinite plays, we consider *leafless* trees and forests, hence the last point of the following definition.

Definition 2. Given a tree alphabet Υ, a Υ-*tree* τ, or *tree* for short when Υ is clear from the context, is a set of words $\tau \subseteq \Upsilon^+$ such that:

1. there is a one symbol word $r = \tau \cap \Upsilon$, called the *root*, such that $r \preccurlyeq x$ for all $x \in \tau$,
2. if $x \cdot d \in \tau$ and $x \neq \epsilon$, then $x \in \tau$, and
3. if $x \in \tau$ then there exists $d \in \Upsilon$ such that $x \cdot d \in \tau$.

A Υ-*forest*, or *forest* when Υ is understood, is defined likewise, removing Point 1.

The following notions of nodes, children, parents and branches, that we define for trees, are similar for forests. The elements of a tree τ are called *nodes*. If $x \neq \epsilon$ and $x \cdot d \in \tau$, we say that $x \cdot d$ is a *child* of x, and that x is the *parent* of $x \cdot d$. We will write $x \cdot \uparrow$ for the parent of a node x: $(x \cdot d) \cdot \uparrow = x$. Note that the root has no parent. The *arity* of a node x, written arity(x), is the number of children of x. If all nodes have arity at most k, then τ is a k-ary tree. Note that Υ-trees are $|\Upsilon|$-ary trees.

A *branch* is an infinite sequence $\lambda = x_0 x_1 \dots$ of nodes such that for all i, x_{i+1} is a child of x_i. For a branch $\lambda = x_0 x_1 \dots$ and an integer $i \geq 0$, $\lambda[i] = x_i$ is the i-th node on the branch, and λ^i denotes the i-th suffix $x_i x_{i+1} x_{i+2} \dots$ Given a node x of a tree τ, we let $Branches(x)$ be the set of branches $\lambda = x_0 x_1 \dots$ that start in node x, *i.e.* such that $x_0 = x$. $Branches(\tau)$ is the set of all branches in τ.

We classically allow nodes of trees and forests to carry additional information. Given a labelling alphabet Σ and a tree alphabet Υ, a Σ-*labelled* Υ-*tree*, or (Σ, Υ)-*tree* for short, is a pair $t = (\tau, \ell)$, where τ is a Υ-tree and $\ell : \tau \to \Sigma$ is a *labelling*. For a node $x = d_1 d_2 \dots d_n$ in τ, we define its *node word* $w(x)$ made of the sequence of labels from the root to this node: $w(x) = \ell(d_1)\ell(d_1 d_2) \dots \ell(d_1 \dots d_n)$. The notion of (Σ, Υ)-*forest* $\mathcal{U} = (u, \ell)$ is defined likewise. Note that we use forests to represent the universe in the semantics of $\mathcal{L}_{\rightsquigarrow}$ (see Section 3.1.1), hence the notations $\mathcal{U}$ and u.

We finish this section by defining, given a forest and a node in the forest, the tree to which this node belongs, *i.e.* the set of nodes in the forest that have the same root.

Definition 3. Let u be a Υ-forest, and let $x = d_1 \dots d_n$ be a node of u. We define the tree u_x as the "greatest" tree in the forest u that contains the node x: $u_x = \{y \in u \mid d_1 \preccurlyeq y\}$. Similarly, given a (Σ, Υ)-forest $\mathcal{U} = (u, \ell)$ and a node $x \in u$, $\mathcal{U}_x = (u_x, \ell_x)$, where u_x is as above and ℓ_x is the restriction of ℓ to the tree u_x.

Note that u_x verifies Point 1 of Definition 2, hence it is indeed a tree.

2.3 Two-player games

We present the notions of arenas, plays and strategies for the classic framework of two-player turn-based games played on graphs.

Arenas

A *game arena*, or *arena* for short, is a tuple $\mathcal{G} = (V, E, V_\iota, v_\iota)$, with a set of *positions* $V = V_1 \uplus V_2$ partitioned between positions belonging to Player 1 (V_1) and those belonging to Player 2 (V_2). $E \subseteq V \times V$ is a relation between positions, describing the possible *moves* between positions, $V_\iota \subseteq V$ is a set of *starting positions* and $v_\iota \in V_\iota$ is the *initial position*.

V_ι has no role in the dynamics of the game, it rather is a lever to tune the range of our full quantifier, as explained in Section 3.1.2. We may omit it from the description of an arena when it is not relevant. We will assume that in every position $v \in V$ there is at least one possible move, *i.e.* $E(v) \neq \emptyset$. We say that Player i *owns* a position v if $v \in V_i$ ($i \in \{1, 2\}$). Also, when there is no ambiguity we may write $v \to v'$ instead of $v\, E\, v'$.

We will often consider game arenas with additional information attached to positions, hence the following definition. Given an alphabet Σ, a Σ-*labelled game arena*, is a tuple $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$, where (V, E, V_ι, v_ι) is a game arena and $\mu : V \to \Sigma$ is a labelling function. μ is extended naturally to sequences of positions: $\mu(v_1 \dots v_n) = \mu(v_1) \dots \mu(v_n)$.

Concretely, in this work, the alphabet will be the set of possible valuations over some finite set of atomic propositions AP, *i.e.* $\Sigma = 2^{AP}$. The propositions in AP represent the relevant information for the uniformity properties one wants to state. If the game models interacting systems, this information can be the values of some (Boolean) variables, or the state of some communication channels. In games with imperfect information, it can be the current observation or what action has just been played.

At last, we define the *size* of a labelled game arena as its number of moves: $|\mathcal{G}| = |E|$.

Plays and paths

In a (labelled or unlabelled) arena, the player owning the initial position v_ι chooses a next position v such that $v_\iota \to v$, then it is to the player owning v to choose an accessible next position, and this process continues for ever, forming an infinite sequence of positions called a *play*. Formally, we define the set of (infinite) plays $Plays_\omega \subseteq V^\omega$ as the set of infinite words $\pi = v_0 v_1 \dots$ such that $v_0 = v_\iota$ and, for each $i \geq 0$, $v_i \to v_{i+1}$. We also define the notion of *partial play*: a finite sequence of positions $\rho = v_0 v_1 \dots v_n$ is a partial play if it is the prefix of a play, and we note $Plays_*$ the set of all partial plays.

Finally, since we will be led to consider sequences of positions that do not start in the initial position of the arena, we define for each position v the sets $Paths_\omega(v)$ and $Paths_*(v)$. They are respectively the set of infinite paths and finite paths starting in v, and are defined like we defined $Plays_\omega$ and $Plays_*$, except that the first position has to be v instead of v_ι. In particular, $Plays_* = Paths_*(v_\iota)$ and $Plays_\omega = Paths_\omega(v_\iota)$. We also define, for $V' \subseteq V$, $Paths_*(V') = \cup_{v \in V'} Paths_*(v)$ and $Paths_\omega(V') = \cup_{v \in V'} Paths_\omega(v)$ as the sets of all finite and infinite paths starting in V'. $Paths_* = Paths_*(V)$ and $Paths_\omega = Paths_\omega(V)$ are thus the sets of all possible (respectively, finite and infinite) paths in the arena. For an infinite path $\pi = v_0 v_1 \dots$ and an integer $i \geq 0$, we use the two following notations: $\pi[i] := v_i$ is the i-th position of the path, and $\pi[0, i] = v_0 \dots v_i$ is the i-th prefix of π; we use similar notations for finite paths.

Strategies

We define two notions of strategies, deterministic and nondeterministic, and a few other notions related to strategies.

Deterministic strategies: A *deterministic strategy* for Player i is a partial function $\sigma : Plays_* \to V$ that maps a partial play ending in a position of Player i to some accessible position: for every partial play ρ such that $\text{last}(\rho) \in V_i$, $\text{last}(\rho)\, E\, \sigma(\rho)$. A play *follows* or

is *induced* by a deterministic strategy for Player i if in this play, every time it is Player i's turn to play, she chooses the next position as prescribed by the strategy. Formally, given a strategy σ for Player i, a play $\pi \in Plays_\omega$ follows, or is induced by σ, if for all $j \geq 0$ such that $\pi[j] \in V_i$, $\pi[j+1] = \sigma(\pi[0,j])$. A deterministic strategy is *memoryless* if its definition only depends on the last position of partial plays.

Nondeterministic strategies: Instead of assigning a unique possible move in each situation, a *nondeterministic strategy*, or *generalized strategy* proposes several possibilities. We define such a strategy for Player i as a partial function $\sigma : Plays_* \to 2^V \setminus \{\emptyset\}$, that maps each partial play ending in a position of Player i to a nonempty set of next positions allowed by the arena: for every partial play ρ such that $\text{last}(\rho) \in V_i$, $\sigma(\rho) \subseteq E(\text{last}(\rho))$. A play $\pi \in Plays_\omega$ is *induced* by a nondeterministic strategy σ for Player i if, for all $j \geq 0$ such that $\pi[j] \in V_i$, $\pi[j+1] \in \sigma(\pi[0,j])$.

Remark 1. Most of the time in this thesis we consider deterministic strategies. As a result we consider understood that, when not otherwise specified, "strategy" means "deterministic strategy". Also all our decision procedures have exactly the same complexity for deterministic and nondeterministic strategies, and with the same techniques exactly, therefore we will only mention it for our first decision procedure (see Remark 8, page 59).

Outcomes: The *outcome* of a (deterministic or generalized) strategy σ, noted $\text{Out}(\sigma) \subseteq Plays_\omega$, is the set of all (infinite) plays that are induced by σ.

Remark 2. In this work, our main concern will be to synthesize strategies whose outcomes verify some properties. For this reason, and because the way a strategy is defined on partial plays that are not induced by this strategy does not affect the outcome, we do not require strategies to be defined on these plays. The kind of strategies we consider thus corresponds to what is often referred to as *plans* in classic game theory.

Finite memory strategies: In general, for a player to follow a strategy she may have to remember arbitrarily long sequences of positions as a play goes on. We will sometimes be interested in strategies that can be implemented with finite memory capabilities.

Definition 4. A *memory structure* is a tuple $\mathfrak{M} = (\Sigma, M, \delta, m_\iota)$ where Σ is a finite set of *events*, M is a set of *memory states*, m_ι is the initial memory state, and $\delta : M \times \Sigma \to M$ is a total function called *memory update function*, which is extended to finite sequences of events the usual way, *i.e.* $\delta(m, \epsilon) = m$ and $\delta(m, e_1 \dots e_{n+1}) = \delta(\delta(m, e_1 \dots e_n), e_{n+1})$.

Definition 5. Let $\mathcal{G} = (V, E, V_\iota, v_\iota)$ be a game arena, and let σ be a deterministic strategy for Player 1. σ is a *finite memory strategy* if there is:

- a *finite* memory structure $\mathfrak{M} = (V, M, \delta, m_\iota)$ over the set of positions, and
- a mapping $\sigma_{\mathfrak{M}} : M \times V \to V$,

such that for every partial play $\rho = \rho' \cdot v$ in the domain of σ, letting $m = \delta(m_\iota, \rho')$, it holds that $\sigma(\rho) = \sigma_{\mathfrak{M}}(m, v)$.

Note that if $\mathfrak{M}$ can be chosen with only one memory state, then σ is a memoryless strategy.

The definition of a *finite memory generalized strategy* is the same, except that the mapping $\sigma_{\mathfrak{M}}$ has $2^V \setminus \{\emptyset\}$ as codomain instead of V.

Strategies as trees: It will often be convenient to see a finite path in a game as a node in a tree, a strategy as a tree and, more generally, the set of finite paths as a forest. To make this correspondence clear, take a *finite* Σ-labelled game arena $\mathcal{G} = (\Sigma, V, E, V_\iota, v_\iota, \mu)$ and a position $v \in V$. Observe that the set $Paths_*(v)$ is a V-tree rooted in v. Also, letting $\ell : x \cdot v' \mapsto \mu(v')$ denote its natural labelling inherited from the arena, $(Paths_*(v), \ell)$ is a Σ-labelled V-tree. Because the labelling of the tree is implicitly given by the labelling of the arena, we will often omit it and see directly $Paths_*(v)$ as a labelled tree. Similarly, for $V' \subseteq V$, $Paths_*(V')$ can be seen as a (Σ, V)-forest that contains one tree per starting position in V', and each finite path ρ in $Paths_*(V')$ is a node in the forest; also, observe that its node word is its sequence of labels: $w(\rho) = \mu(\rho)$. In the same fashion, a strategy for Player i can be seen as a subtree of $Plays_*$ (seen as a (Σ, V)-tree) obtained by pruning some moves of Player i. Formally, a *deterministic strategy tree* $t = (\tau, \ell)$ for Player i is a Σ-labelled V-tree with root v_ι such that for every $x \in \tau$, letting $v = \mathrm{last}(x)$, it holds that:

1. if $v \in V_i$, then x has a unique child $x \cdot v'$ with $v' \in E(v)$, and $\ell(x \cdot v') = \mu(v')$
2. if $v \in V_{3-i}$, then x has one child $x \cdot v'$ for each $v' \in E(v)$, and $\ell(x \cdot v') = \mu(v')$.

Every deterministic strategy σ defines a unique deterministic strategy tree that we shall write t_σ.

In the case of nondeterministic strategies, the definition of *nondeterministic strategy trees* is the same as for deterministic strategy trees, except that in Point 1, x can have more than one child, representing the possible next positions allowed by the strategy.

Remark 3. Strategy trees correspond exactly to strategies that are defined only on plays induced by the strategy, the relevance of which is justified in Remark 2, page 15.

Parity games and memoryless determinacy

A *parity game* $G = (\mathcal{G}, C)$ is a game arena $\mathcal{G} = (V, E, V_\iota, v_\iota)$ together with a *colouring function* $C : V \to \mathbb{N}$ with finite codomain, that assigns a *colour* or *priority* to each position. A play π is *winning* for Player 1 if the least priority seen infinitely often in π is even. Otherwise it is winning for Player 2.

Given a parity game $G = (\mathcal{G}, C)$, a strategy σ for Player i is a *winning strategy* if Player i wins all the plays induced by σ.

A game is said to be *determined* if one of the two players has a winning strategy. We state a fundamental result of parity games played on (possibly infinite) graphs.

Theorem 1 (Zielonka (1998))**.** *Every parity game is determined, and the player that has a winning strategy has a memoryless one.*

Solving a two-player game means to decide whether Player 1 has a winning strategy. In case the game graph is finite, the following complexity bound is well known.

Theorem 2 (Emerson and Lei (1986); McNaughton (1993); Zielonka (1998))**.** *Solving a parity game with n positions, m edges and l colours can be done in time $O(m \cdot n^l)$.*

2.4 Logics

Throughout this work, AP will denote a finite nonempty set of *atomic propositions*. Though we will always use only a finite number of such propositions in each instance of the problems we consider, this number cannot be bounded. For this reason we do not fix it once and for all, but we will always assume that it is big enough to contain all the atomic propositions needed.

2.4.1 Propositional logic

We start with basic propositional logic, or boolean logic.

Syntax

To distinguish propositional logics' formulas from other logics' formulas, we denote them by $\alpha, \beta \ldots$ For a set of atomic propositions AP, the syntax of propositional logic is defined by the following grammar:

$$\alpha ::= p \mid \neg\alpha \mid \alpha \vee \alpha,$$

where p is in AP. The set of well-formed boolean formulas over AP is denoted by $\mathbb{B}(AP)$. As usual, we define the following abbreviations: $\alpha \wedge \beta := \neg(\neg\alpha \vee \neg\beta)$, $\alpha \to \beta := \neg\alpha \vee \beta$, $\alpha \leftrightarrow \beta := \alpha \to \beta \wedge \beta \to \alpha$, $\texttt{true} := p \vee \neg p$ and $\texttt{false} := p \wedge \neg p$ for some $p \in AP$.

Semantics

A boolean formula is evaluated in a *valuation* for the atomic propositions. A valuation $\nu \subseteq AP$ is a subset of the atomic propositions, with the meaning that the propositions in ν are true, and the others are false.

The *satisfaction* of a boolean formula α by a valuation ν, noted $\nu \models \alpha$, is defined by induction over formulas: $\nu \models p$ if $p \in \nu$, $\nu \models \neg\alpha$ if it is not true that $\nu \models \alpha$ and $\nu \models \alpha \vee \beta$ if $\nu \models \alpha$ or $\nu \models \beta$.

2.4.2 First-order logic

We recall the syntax and semantics of first-order logic. Concerning semantics we not only give the classic Tarski's semantics, but also the game semantics, as in Section 3.3 we will consider some variants of these games, in the context of logics of imperfect information.

Syntax

First-order logic is basically propositional logic with, instead of atomic propositions, relations or *predicates* over *variables* that range over some *domain*, and with the possibility of quantifying over those variables. Note that the full syntax of first-order logic allows for constant and function symbols, but since for each formula that uses them there is an equivalent formula that does not, and because they are not relevant for our study, we choose to present the simple version that uses only variables and predicates. Let *Var* be some set of variable symbols.

The syntax is as follows:

$$\varphi ::= R(x_1, \ldots, x_n) \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x\varphi,$$

where R is a n-ary relation symbol, and $x, x_1, \ldots, x_n$ are variable symbols in *Var*.

For a first-order formula φ, we define *Sub*(φ) as the set of subformulas of φ. We say that a variable x is *free* in a formula φ if there is at least one occurence of x in φ that is not bounded by a quantifier. We let *Free*(φ) be the set of free variables in φ. Also, we let *Var*(φ) be the set of variables that appear in φ.

Tarski's semantics

To give a truth value to a first-order formula, we first need a set of possible values for the variables, and an interpretation for each relation symbol that appears in the formula. Formally an *interpretation* or *model* for a formula is a pair $\mathcal{M} = (D, I)$, where D is a nonempty set of values called the *domain*, and I is a function that maps each n-ary relation symbol R of the formula to an actual n-ary relation $I(R) \subseteq D^n$.

Then, if there are free variables in the formula, we also need to give them a value. Given a formula φ and a model $\mathcal{M} = (D, I)$, a *variable assignment* for φ is a partial function $s : \mathit{Var}(\varphi) \to D$, that must be defined at least on *Free*(φ). Given an assignment s for φ, a variable $x \in \mathit{Var}(\varphi)$ and a value $a \in D$, $s[a/x] : \mathit{Var}(\varphi) \to D$ is the assignment that maps x to a, and is equal to s on $\mathit{Var}(\varphi) \setminus \{x\}$.

Now, given a formula φ, a model $\mathcal{M} = (D, I)$ and an assignment s for φ, we can inductively define the truth value of φ in the model $\mathcal{M}$ with assignment s:

$\mathcal{M}, s \models R(x_1, \ldots, x_n)$ if $(s(x_1), \ldots, s(x_n)) \in I(R)$
$\mathcal{M}, s \models \neg\varphi$ if it is not the case that $\mathcal{M}, s \models \varphi$
$\mathcal{M}, s \models \varphi \vee \psi$ if $\mathcal{M}, s \models \varphi$ or $\mathcal{M}, s \models \psi$
$\mathcal{M}, s \models \exists x\varphi$ if there is $a \in D$ such that $\mathcal{M}, s[a/x] \models \varphi$

Game semantics

An equivalent way of defining the truth of a formula is by means of a game played between two players, the Verifier and the Spoiler. Given a formula Φ, a model $\mathcal{M} = (D, I)$ and an assignment s_ι for Φ, we define the game arena $\mathcal{G}^{\Phi}_{\mathcal{M},s} = (V, E, v_\iota)$, where the set of positions is $V = \mathit{Sub}(\Phi) \times (\mathit{Var}(\Phi) \to D) \times \{\mathit{Ve}, \mathit{Sp}\}$ and the initial position is $v_\iota = (\Phi, s_\iota, \mathit{Ve})$. In a position (φ, s, X), if $X = \mathit{Ve}$ (resp. $X = \mathit{Sp}$), then Verifier (resp. Spoiler) is currently in charge of proving the subformula φ to be true in the model $\mathcal{M}$ with variable assignment s. When φ is a disjunction, X chooses one of the disjuncts; if it is an existential quantification on variable x, she chooses a value in D for x, and if it is a negation then the player in charge changes. A play ends when an atomic formula is reached. Formally, the moves are as follows. In a position (φ, s, X), if

$\varphi = R(x_1, \ldots, x_n)$, X wins if $(s(x_1), \ldots, s(x_n)) \in I(R)$
$\varphi = \neg\psi$, the next position is (ψ, s, X'), where X' is the opponent of X
$\varphi = \psi_1 \vee \psi_2$, X chooses a subformula ψ_i and moves to (ψ_i, s, X) $(i \in \{1, 2\})$
$\varphi = \exists x\psi$, X chooses a value $a \in D$ and moves to $(\psi, s[a/x], X)$.

Proposition 1. *For a first-order formula φ, a model $\mathcal{M}$ and an assignment s, $\mathcal{M}, s \models \varphi$ if and only if Verifier has a winning strategy in $\mathcal{G}^{\varphi}_{\mathcal{M},s}$.*

Note that positions associated with atomic formulas have no successor, and all plays are finite, which does not match the definition of games given in Section 2.3. However it is straightforward to transform these semantic games into games of infinite duration with a very simple parity condition.

For all the logics that we consider, the size of a formula φ, noted $|\varphi|$, is defined as its number of symbols.

2.5 Automata

For an alphabet Σ, we call a subset of Σ^* a *word language*, and a subset of Σ^ω is an *ω-word language*. Similarly, for a tree alphabet Υ and a labelling alphabet Σ, a set of (Σ, Υ)-trees is an *ω-tree language*. We recall the fundamental notion of word automata, both for finite and infinite words, and we also define nondeterministic, alternating and two-way automata on infinite trees.

2.5.1 Automata on finite words

A *nondeterministic word automaton* is a tuple $\mathcal{A} = (\Sigma, Q, \Delta, q_\iota, F)$, where Σ is an alphabet, Q is a finite set of *states*, $\Delta \subseteq Q \times \Sigma \times Q$ is a *transition relation* and F is a set of *accepting* states. If $\Delta(q, a)$ is nonempty for every $q \in Q$ and $a \in \Sigma$ we say that $\mathcal{A}$ is *complete*. A *run* of $\mathcal{A}$ over a word $w = a_1 \dots a_n \in \Sigma^*$ is a sequence of states $q_0 \dots q_n$ such that $q_0 = q_\iota$ and for each $i \in \{0, \dots, n-1\}$, $(q_i, a_{i+1}, q_{i+1}) \in \Delta$. A run $q_0 \dots q_n$ is *accepting* if $q_n \in F$, and we say that a word w is *accepted* by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ over w. The *language* accepted by a word automaton $\mathcal{A}$ consists in the set of words accepted by $\mathcal{A}$, and it is written $\mathcal{L}(\mathcal{A})$. It is well known that the set of languages accepted by nondeterministic word automata is precisely the set of *regular* word languages. We shall also use the notion of *trace*: a word w is a trace of the automaton $\mathcal{A}$ if there is a run, not necessarily accepting, of $\mathcal{A}$ over w. We will note $\mathcal{T}(\mathcal{A})$ the set of traces of a word automaton $\mathcal{A}$.

When the transition relation of a word automaton $\mathcal{A}$ verifies $|\Delta(q, a)| \leq 1$ for each state q and letter a, we say that $\mathcal{A}$ is *deterministic*. In this case we may represent the transitions of the automaton by means of a partial function $\delta : Q \times \Sigma \to Q$ instead of a relation Δ: for each state q and letter a, if $\Delta(q, a) = \{q'\}$ then we let $\delta(q, a) = q'$, and if $\Delta(q, a) = \emptyset$ then $\delta(q, a)$ is left undefined. We will also use the classic extension of δ to words by letting, for each state q, $\delta(q, \epsilon) = q$ and, for each word w and letter a, $\delta(q, w \cdot a) = \delta(\delta(q, w), a)$ when this is defined. Since every nondeterministic word automaton can be turned into an equivalent deterministic one (when we talk of equivalent automata we refer to language equality), we have the classic result that deterministic automata also accept the set of regular word languages. For an introduction to word automata and regular languages, see Hopcroft et al. (2003).

2.5.2 Automata on infinite words

The notion of word automaton is extended to the case of infinite words by introducing more involved acceptance conditions suited for infinite runs. Several such acceptance conditions exist: Büchi, co-Büchi, Rabin, Streett, Muller and parity. For nondeterministic word automata they all define the same set of ω-word languages, namely ω-regular word languages. In this work we shall only use the parity condition.

A *parity word automaton* is a tuple $\mathcal{A} = (\Sigma, Q, \Delta, q_\iota, C)$, where Σ, Q, Δ and q_ι are as for nondeterministic finite word automata, and $C : Q \to \mathbb{N}$ is a colouring function assigning an integer called *colour* or *priority* to each state of the automaton. Runs of parity word automata are defined as for finite word automata, except that they are here infinite sequences of states. A run is said to be *accepting* if the least priority seen infinitely often during this run is even. The notions of acceptance of a word and accepted languages are similar to the case of finite word automata.

Again, a parity word automaton is *deterministic* if, given a current state and a letter, the transition relation allows at most one successor state. For all acceptance conditions except Büchi and co-Büchi, deterministic infinite word automata accept the same ω-languages as nondeterministic ones. Thomas (1990) is a good introduction to the theory of automata on infinite words.

2.5.3 Automata on infinite trees

Let Υ be a tree alphabet and let Σ be a labelling alphabet. We recall the notions of nondeterministic, alternating, and two-way alternating tree automata on infinite Σ-labelled Υ-trees. For an introduction to the theory of automata on infinite trees see Thomas (1990) for nondeterministic tree automata, Löding (2014) for alternating tree automata, and Vardi (1998) for two-way alternating tree automata.

First, note that automata on infinite trees use the same acceptance conditions as automata on infinite words (Büchi, co-Büchi, Rabin, Streett, Muller and parity). However, these conditions do not all define the same set of ω-tree languages, as they do for ω-word languages. Indeed, while Rabin, Street, Muller and parity tree automata all recognize the same ω-tree languages, which are the *ω-regular tree languages*, Büchi and co-Büchi tree automata recognize strictly less languages, and in particular they are not closed for complementation.

In this work we only consider parity acceptance condition, as Büchi condition is not enough for our purposes, and parity is more convenient to handle than Street, Rabin or Muller conditions.

Definition of tree automata

For a set X, $\mathbb{B}^+(X)$ is the set of positive boolean formulas over X, *i.e.* formulas built with elements of X as atomic propositions and using only connectives $\vee$ and $\wedge$. As usual, we also allow for formulas `true` and `false`, and $\wedge$ has precedence over $\vee$. Elements of $\mathbb{B}^+(X)$ are denoted by $\alpha, \beta \dots$

Definition 6. Let $Dir \subseteq \Upsilon \cup \{\epsilon, \uparrow\}$ be a set of *transition directions*. A *Dir-parity tree automaton*, or *Dir*-automaton for short, is a tuple $\mathcal{A} = (\Sigma, Q, \delta, q_\iota, C)$ where Σ is a finite

alphabet, Q is a finite set of states, $q_\iota \in Q$ is an initial state, $C : Q \to \mathbb{N}$ is a colouring function, and $\delta : Q \times \Sigma \to \mathbb{B}^+(Dir \times Q)$ is a transition function.

We define the size of a Dir-automaton $\mathcal{A} = (\Sigma, Q, \delta, q_\iota, C)$ as the sum of the sizes of formulas in its transition function, plus the number of colours used in the acceptance condition:

$$|\mathcal{A}| = \sum_{q \in Q} \sum_{a \in \Sigma} |\delta(q, a)| + |C(Q)|.$$

Let $Dir_A = \Upsilon$ and $Dir_\uparrow = Dir_A \cup \{\epsilon, \uparrow\}$ denote the set of transition directions used respectively by alternating and two-way alternating tree automata.

Definition 7. Dir_A-automata are *alternating tree automata* and $Dir_\uparrow$-automata are *two-way alternating tree automata*, or two-way tree automata for short.

Before formally defining the acceptance of a tree by an automaton, we informally describe the behaviour of a Dir-automaton in a state q reading a node of a tree labelled with a. Recall that the transition function gives a formula $\delta(q, a) \in \mathbb{B}^+(Dir \times Q)$. The automaton has to choose a (possibly empty) set of pairs $\{[d_1, q_1], \ldots, [d_n, q_n]\}$, where for each i, $d_i \in Dir$ and $q_i \in Q$, such that this set seen as a valuation over $Dir \times Q$ verifies $\delta(q, a)$. Then, for each $1 \leq i \leq n$, the automaton sends a copy of itself in direction d_i in the input tree, where it continues in state q_i. So while the automaton is executed on a tree, it may actually spawn many parallel executions, and all these executions must be accepting, according to the parity acceptance condition, for the input tree to be accepted. For example, if $\Upsilon = \{1, 2\}$, consider that the transition of a two-way tree automaton in state q, when reading a, is $\delta(q, a) = [1, q_1] \wedge [\uparrow, q_2] \vee [1, q_3] \wedge [1, q_2]$. It means that the two-way automaton has two options. The first one is to send to the first child of the current node a copy of itself in state q_1, and send to the parent of the current node a copy of the automaton in state q_2. The second option is to send two copies of itself to the first child, one in state q_3 and one in state q_2.

Two-way tree automata can send copies of themselves in all directions, including the parent of the current node and the current node itself, thus staying in place. Alternating tree automata can only send copies to the children of the current node, but there is no further restriction on how these copies are sent. In particular, an alternating tree automaton can send several copies of itself in different states to the same child of the current node, thus launching several different executions on the same subtree.

Nondeterministic tree automata are obtained by forbidding this kind of behaviour. The transitions of such automata are classically given by a relation $\Delta \subseteq Q \times \Sigma \times Q^{|\Upsilon|}$, and we shall sometimes adopt this vision. [1] For a nondeterministic automaton, we call a *transition* an element of its transition relation.

In the formalism of Definition 6 however, an alternating tree automaton is nondeterministic if each formula in δ is a disjunction of conjunctions of atoms, such that each disjunct has exactly one atomic conjunct for each direction in Υ. A disjunct thus represents a possible transition, and each transition assigns exactly one state to each child of the current node. For example, if $\Upsilon = \{1, 2\}$, $\delta(q, a) = [1, q_1] \wedge [2, q_2] \vee [1, q_2] \wedge [2, q_1]$

1. In this case we assume that the finite tree alphabet Υ is ordered.

can be part of the transition function of a nondeterministic tree automata, while $\delta(q, a) = [1, q_1] \wedge [2, q_2] \vee [1, q_2] \wedge [1, q_1]$ cannot, as the second disjunct sends two copies of the automaton to the same child.

Enforcing several different executions to take place *and succeed* when reading an input can be seen as *universal choices*, dual to *existential choices* of nondeterminism. The alternation between existential and universal choices in alternating tree automata makes it very natural to define the acceptance of a tree by means of two-player games, as we do below.

Note that in most works, tree automata take as input labellings of the *complete* input tree Υ^*. In Vardi (1995), the transition function depends on the arity of the current node, and it forbids to use in the transition formulas directions in which there is no child. Instead, we allow the automaton to try and send a copy in some state q in a direction where there is no node. It should just be specified for each state q whether this behaviour is accepted or rejected (take two copies of each state if necessary). Our automata can then be simulated by standard automata on the complete tree Υ^*. A Σ-labelled Υ-tree is represented by a $\Sigma \cup \{\bot\}$-labelling of the complete tree Υ^*, where $\bot$ represents absent nodes. The transition relation is just slightly modified such that when in a state q, if the current node of the complete tree is labelled by $\bot$, then the automaton sends copies in state $q_\bot$ in all directions. $q_\bot$ is accepting or rejecting depending on what was specified for q. If $q_\bot$ is rejecting it rejects immediately, and if it is accepting, it accepts provided all the subtree is also labelled by $\bot$.

The previous discussion is due to the possibility for tree automata to send copies of themselves in specific directions. However, most of the time in this work, the ability to identify the precise child of the current node in which an automaton must send a copy is unnecessary, and specifying instead that the automaton must send a copy in at least one child or in all children is enough. Therefore, because it simplifies the presentation, and unless otherwise mentioned, we replace the set of transition directions Dir_A for alternating tree automata with the set of *abstract directions* $\{\Diamond, \Box\}$. $[\Diamond, q]$ means that the automaton nondeterministically chooses a child of the current node to which it sends a copy in state q, while $[\Box, q]$ sends a copy in state q to all the children of the current node. To illustrate the discussion of the previous paragraph, see that $[\Diamond, q]$ could be replaced by $\bigvee_{d \in \Upsilon} [d, q]$, where q rejects when sent to an unexisting node, and dually $[\Box, q]$ has the same meaning as $\bigwedge_{d \in \Upsilon} [d, q]$, where q accepts when sent to an absent node. Such abstract directions in alternating automata have been used to define alternating automata working on graphs (Bojanczyk, 2002; Piterman and Vardi, 2004). When studying logics that do not distinguish the different successors of a node, like CTL^*, such abstract directions are also sometimes considered; in Kupferman et al. (2000) for example, alternating tree automata using these abstract directions are called *symmetric*.

Acceptance game

As announced, and following a classic approach (Muller and Schupp, 1987; Löding, 2014), acceptance of a tree by a *Dir*-automaton is defined on a two-player parity game between Eve (the proponent) and Adam (the opponent). Let $t = (\tau, \ell)$ be a (Σ, Υ)-tree, let $x_\iota \in \tau$, and let $\mathcal{A} = (\Sigma, Q, \delta, q_\iota, C)$ be a *Dir*-automaton for some set of transition directions *Dir*. We define the parity game $\mathcal{G}^{x_\iota}_{\mathcal{A},t} = (V, E, v_\iota, C')$: the set of positions is

$V = \tau \times Q \times \mathbb{B}^+(Dir \times Q)$, the initial position is $(x_\iota, q_\iota, \delta(q_\iota, x_\iota))$, and a position (x, q, α) belongs to Eve if α is of the form $\alpha_1 \vee \alpha_2$ or $[\Diamond, q']$; otherwise it belongs to Adam. Moves in $\mathcal{G}^{x_\iota}_{\mathcal{A},t}$ are defined by the following rules; for clarity we shall abuse notations, and for a node x of t and a state q of $\mathcal{A}$ we write $\delta(q, x)$ for $\delta(q, \ell(x))$.

$$
\begin{array}{ll}
(x, q, \alpha_1 \dagger \alpha_2) \to (x, q, \alpha_i) & \text{where } \dagger \in \{\vee, \wedge\} \text{ and } i \in \{1, 2\}, \\
(x, q, [\bigcirc, q']) \to (y, q', \delta(q', y)) & \text{where } \bigcirc \in \{\Diamond, \Box\} \text{ and } y \text{ is a child of } x, \\
(x, q, [\epsilon, q']) \to (x, q', \delta(q', x)) & \text{and} \\
(x, q, [\uparrow, q']) \to (y, q', \delta(q', y)) & \text{where } y \text{ is } x\text{'s parent.}
\end{array}
$$

These moves read as follows. Because Eve owns positions of the form $(x, q, \alpha_1 \vee \alpha_2)$ and Adam owns those of the form $(x, q, \alpha_1 \wedge \alpha_2)$, the first rule means that Eve chooses the disjuncts and Adam chooses the conjuncts in the transition formula. Note that neither the current node in the tree nor the state of the automaton changes while an atom is not reached. Again, because Eve owns positions of the form $(x, q, [\Diamond, q'])$ and Adam those of the form $(x, q, [\Box, q'])$, the second rule means that when $\bigcirc = \Diamond$ (resp. $\bigcirc = \Box$), Eve (resp. Adam) chooses a child to which the automaton sends a copy in state q' (recall that in our definition of trees, a node always has at least one child). Concerning the two last rules, positions of the form $(x, q, [\epsilon, q'])$ and $(x, q, [\uparrow, q'])$ all belong to Adam, but this is arbitrary. In the first case, the only possible move is to stay in place and switch to state q', while in the second case the only possible move is to send the state q' to the parent node.

Positions of the form $(x, q, \texttt{true})$ and $(x, q, \texttt{false})$ are sink positions [2], winning for Eve and Adam respectively. Positions of the form $(r, q, [\uparrow, q'])$ where r is the root of the input tree are also sink positions as the root of a tree has no parent; they are winning for Adam. The colouring is inherited from the one of the automaton: $C'(x, q, \alpha) = C(q)$, except for sink positions, which are assigned an even (resp. odd) priority if they are winning for Eve (resp. Adam).

A tree t rooted in r is *accepted* by $\mathcal{A}$ if Eve has a winning strategy in $\mathcal{G}^r_{\mathcal{A},t}$, and we denote by $\mathcal{L}(\mathcal{A})$ the set of trees accepted by $\mathcal{A}$.

In case concrete directions $Dir_A = \Upsilon$ are allowed (instead of abstract directions $\Diamond$ and $\Box$), then the evaluation game $\mathcal{G}^{x_\iota}_{\mathcal{A},t}$ is modified as follows. Positions of the form $(x, q, [d, q])$, where $d \in \Upsilon$, have only one possible next position and arbitrarily belong to Adam. The corresponding moves are:

$$(x, q, [d, q']) \to (x \cdot d, q', \delta(q', x \cdot d)) \quad \text{if } x \cdot d \in t.$$

If $x \cdot d \notin t$, then $(x, q, [d, q'])$ is a sink position. It is either accepting or rejecting depending on q' (recall the discussion of Page 22).

Simulation theorems

The most important result about alternating tree automata is that they can be simulated by nondeterministic tree automata. This comes at the price of an exponential blowup in the number of states, however the number of parities only grows linearly.

2. It is more common to let them be deadlocks, but we equivalently make them sink positions to fit our definition of game arenas.

Theorem 3 (Muller and Schupp (1995))**.** *Let $\mathcal{A}$ be an alternating parity tree automaton with n states and l colours. There is a nondeterministic parity tree automaton that accepts the same language, and has no more than $2^{O(n \cdot l \cdot \log(n \cdot l))}$ states and $O(n \cdot l)$ colours.*

A similar result holds for two-way alternating tree automata.

Theorem 4 (Vardi (1998))**.** *Let $\mathcal{A}$ be a two-way alternating parity tree automaton with n states and l colours. There is a nondeterministic parity tree automaton that accepts the same language, and has no more than $2^{O(n \cdot l \cdot \log(n \cdot l))}$ states and $O(n \cdot l)$ colours.*

As a result, nondeterministic, alternating and two-way parity tree automata all define the same set of tree languages, which are the ω-regular tree languages.

Nonemptiness of tree automata

The *nonemptiness problem* consists in deciding whether the language of a given automaton is empty or not. A classic approach to solve this problem for nondeterministic parity tree automata is by means of parity games. The idea is that in each state of the automaton, the first player guesses the labelling of the current node and the transition to take in the automaton, and the second player chooses a direction in which to proceed.

Definition 8. Given a nondeterministic parity tree automaton $\mathcal{A} = (\Sigma, Q, \Delta, q_\iota, C)$ over Σ-labelled Υ-trees, define its *nonemptiness parity game* $\mathcal{G}_\mathcal{A} = (V, E, v_\iota, C')$, where:

- $V = Q \cup \Delta$, $V_1 = Q$, $V_2 = \Delta$,
- if $(q, a, q_1, \dots, q_{|\Upsilon|}) \in \Delta$, then $q \to (q, a, q_1, \dots, q_{|\Upsilon|})$ and for each $i \in \{1, \dots, |\Upsilon|\}$, $(q, a, q_1, \dots, q_{|\Upsilon|}) \to q_i$,
- $v_\iota = q_\iota$, $C'(q) = C(q)$ and $C'((q, a, q_1, \dots, q_{|\Upsilon|})) = C(q)$.

Fact 1. *$\mathcal{L}(\mathcal{A}) \neq \emptyset$ if, and only if, Player 1 has a winning strategy in $\mathcal{G}_\mathcal{A}$.*

Combining Fact 1 with Theorem 2 gives the following complexity bound.

Theorem 5 (Löding (2014))**.** *Let $\mathcal{A}$ be a nondeterministic tree automaton over (Σ, Υ)-trees with n states, m transitions, and l colours. Let $d = |\Upsilon|$ be the maximal arity of trees. The nonemptiness problem for $\mathcal{A}$ can be solved in time $O(d \cdot m \cdot (n + m)^l)$.*

We simplify this expression by remarking that for nondeterministic tree automata with n states working over (Σ, Υ)-trees, the size m of the transition relation $\Delta \subseteq Q \times \Sigma \times Q^{|\Upsilon|}$ is bounded by $|\Sigma| \cdot n^{|\Upsilon|+1}$.

Proposition 2. *Let $\mathcal{A}$ be a nondeterministic tree automaton over (Σ, Υ)-trees with n states and l colours. Let $d = |\Upsilon|$ be the maximal arity of trees. The emptiness problem for $\mathcal{A}$ can be solved in time $(|\Sigma| \cdot n^{O(d)})^{O(l)}$.*

Proof. By Theorem 5, the emptiness problem can be solved in time $O(d \cdot m \cdot (n + m)^l)$, where m is the number of transitions. As observed above, we have $m \leq |\Sigma| \cdot n^{d+1}$. Denoting the time complexity by θ, there is some constant k such that:

$$\begin{aligned}
\theta &\leq k \cdot d \cdot |\Sigma| \cdot n^{d+1} \cdot (n + |\Sigma| \cdot n^{d+1})^l \\
&\leq k \cdot d \cdot |\Sigma| \cdot n^{d+1} \cdot (2 \cdot |\Sigma| \cdot n^{d+1})^l \\
&\leq (2 \cdot k \cdot d \cdot |\Sigma| \cdot n^{d+1})^{l+1} \\
\theta &\leq (|\Sigma| \cdot n^{O(d)})^{O(l)}
\end{aligned}$$

Note that the last inequality only holds if n is strictly greater than 1, but testing the emptiness of an automaton with one state is trivially done in constant time. □

Regular trees

We finish this section with a result crucial for program synthesis, which is that every nonempty ω-regular tree language contains a finitely generated tree.

Definition 9. A (Σ, Υ)-tree $t = (\tau, \ell)$ is *regular* if there is a deterministic finite state word automaton $\mathcal{T} = (\Upsilon, Q, \delta, q_\iota, F)$ and a state labelling $\ell_\mathcal{T} : Q \to \Sigma$ such that $\mathcal{L}(\mathcal{T}) = \tau$, and for every node $x \in \tau$, $\ell(x) = \ell_\mathcal{T}(\delta(q_\iota, x))$.

Theorem 6 (Hossley and Rackoff (1972))**.** *Every nonempty ω-regular tree language contains a regular tree.*

Moreover, classic algorithms for testing the nonemptiness of nondeterministic tree automata are constructive, and in case the language of the tested automaton is not empty, they provide the description of a finite automaton generating some regular tree in the language (see Pnueli and Rosner (1989) for the case of Rabbin tree automata). Concerning parity tree automata, recall that the language of a nondeterministic parity tree automaton $\mathcal{A}$ is nonempty if, and only if, Player 1 has a winning strategy in the nonemptiness parity game $\mathcal{G}_\mathcal{A}$ (Fact 1). Because parity games are memoryless determined (Theorem 1), if Player 1 has a winning strategy then she has a memoryless one. Most algorithms that solve parity games on finite graphs compute such a strategy when it exists. We describe how, from a memoryless winning strategy for Player 1, one can build a finite word automaton and a state labelling that generate a regular tree in $\mathcal{L}(\mathcal{A})$.

Definition 10. Let $\mathcal{A}$ be a nondeterministic parity tree automaton working over (Σ, Υ)-trees, and let $\mathcal{A}^\perp = (\Sigma \cup \{\perp\}, Q, \Delta, q_\iota, C)$ be the equivalent automaton working on the complete tree Υ^*. Recall that the special symbol $\perp$ denotes missing nodes. Let $\mathcal{G}_{\mathcal{A}^\perp} = (V, E, v_\iota, C)$ be the nonemptiness game for $\mathcal{A}^\perp$, and assume that Player 1 has a memoryless winning strategy σ. To simplify presentation assume that $\Upsilon = \{1, \ldots, k\}$. We build the word automaton $\mathcal{T} = (\Upsilon, Q^\mathcal{T}, \delta^\mathcal{T}, q_\iota^\mathcal{T}, F^\mathcal{T})$ and the state labelling $\ell_\mathcal{T}$, where:

- $Q^\mathcal{T} = Q = V_1$,
- $q_\iota^\mathcal{T} = q_\iota$,
- for $q \in Q^\mathcal{T}$ and $d \in \Upsilon$, if $\sigma(q) = (q, a, q_1, \ldots, q_k)$ then $\delta^\mathcal{T}(q, d) = q_d$ and $\ell_\mathcal{T}(q) = a$,
- $F^\mathcal{T} = \{q \mid \ell_\mathcal{T}(q) \neq \perp\}$.

Define $\tau = \mathcal{L}(\mathcal{T})$, and for $x \in \tau$ let $\ell(x) = \ell_\mathcal{T}(\delta^\mathcal{T}(q_\iota^\mathcal{T}, x))$.

Fact 2. *$t = (\tau, \ell)$ is a regular tree, and $t \in \mathcal{L}(\mathcal{A})$.*

2.6 Rational relations

Relations over finite words are infinite objects in general, and manipulating them requires finite representations. Rational languages is a classic example of a class of (potentially) infinite languages of words that can be finitely represented, and the corresponding

notion concerning relations is the one of *rational relations*. The class of rational relations is very rich as we shall see, and like regular languages they are recognized by finite state machines called transducers, which allows to address algorithmic issues involving these relations. We define finite state transducers and recall basic results concerning rational relations as well as some subclasses of interest. Much more information concerning these matters can be found in Berstel (1979).

As in this work we only consider binary rational relations, we will not define transducers for relations of arbitrary arity but just for binary ones. One way to see such a transducer is to picture a nondeterministic automaton with two tapes. Given two input finite words, one on each tape, the automaton reads them according to a set of possible transitions, and when it has reached the end of both words, it accepts the pair if it is in an accepting state. Notice that the transducer in general does not have to progress at the same pace on both tapes.

Another way to see a transducer which will be useful is to consider that the first tape is an *input* tape and the second is an *output* tape. The transducer reads an input finite word on its input tape and writes out a finite word on its output tape. This machine being in general nondeterministic, it may have several outputs for a given input word.

Definition 11. A *Finite State Transducer* (FST) is a tuple $T = (\Sigma, \Gamma, Q, \Delta, q_\iota, F)$, where Σ is an *input alphabet* and Γ an *output alphabet*, Q is a finite set of states, $q_\iota \in Q$ is the *initial state*, $F \subseteq Q$ is a set of *accepting states*, and $\Delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \times Q$ is a finite set of *transitions*. The transducer is called *synchronous* if $\Delta \subseteq Q \times \Sigma \times \Gamma \times Q$.

Intuitively, $(q, a, b, q') \in \Delta$ means that the transducer can move from state q to state q' by reading a and writing b (both possibly ϵ, except for synchronous transducers).

We also define the *extended transition relation* $\Delta^* \subseteq Q \times \Sigma^* \times \Gamma^* \times Q$, which is the smallest relation such that:

- for all $q \in Q$, $(q, \epsilon, \epsilon, q) \in \Delta^*$, and
- if $(q, w, w', q') \in \Delta^*$ and $(q', a, b, q'') \in \Delta$, then $(q, w \cdot a, w' \cdot b, q'') \in \Delta^*$.

For $q, q' \in Q$, $w \in \Sigma^*$ and $w' \in \Gamma^*$, the notation $q \text{-}[w/w']\!\rightarrow q'$ means that $(q, w, w', q') \in \Delta^*$. The *relation recognized by* T is:

$$[T] := \{(w, w') \mid w \in \Sigma^*, w' \in \Gamma^*, \exists q \in F,\ q_\iota \text{-}[w/w']\!\rightarrow q\}.$$

In other words, a pair (w, w') is in the relation recognized by T if there is an accepting execution of T that reads w and writes w'.

We classically define the composition of two transducers:

Definition 12. Let $T_1 = (\Sigma, \Gamma, Q^1, \Delta^1, q^1_\iota, F^1)$ and $T_2 = (\Gamma, \Omega, Q^2, \Delta^2, q^2_\iota, F^2)$ be two FST. The *composition* of T_1 and T_2 is the FST $T_1 \circ T_2 = (\Sigma, \Omega, Q^1 \times Q^2, \Delta, (q^1_\iota, q^2_\iota), F^1 \times F^2)$, where, for $q_1, q'_1 \in Q^1, q_2, q'_2 \in Q^2$, $a \in \Sigma \cup \{\epsilon\}$ and $c \in \Omega \cup \{\epsilon\}$, $((q_1, q_2), a, c, (q'_1, q'_2)) \in \Delta$ if

- there is $b \in \Gamma$ such that $(q_1, a, b, q'_1) \in \Delta^1$ and $(q_2, b, c, q'_2) \in \Delta^2$, or
- $(q_1, a, \epsilon, q'_1) \in \Delta^1$, $c = \epsilon$ and $q'_2 = q_2$, or
- $(q_2, \epsilon, c, q'_2) \in \Delta^2$, $a = \epsilon$ and $q'_1 = q_1$.

The following is folklore:

Fact 3. *For a transducer T_1 over alphabets Σ and Γ and a transducer T_2 over alphabets Γ and Ω, $[T_1 \circ T_2] = [T_1] \circ [T_2]$, and $|T_1 \times T_2| = O(|T_2| \times |T_2|)$.*

Definition 13 (Rational and regular relations)**.** Let Σ and Γ be two alphabets. A binary relation $\rightsquigarrow \subseteq \Sigma^* \times \Gamma^*$ is *rational* if there is a finite state transducer T such that $[T] = \rightsquigarrow$. A binary relation is *regular* if it can be recognized by a synchronous transducer. [3]

Rat and Reg are respectively the set of rational relations and the set of regular relations. We will often consider transducers that have the same alphabet Σ for input and output, and in this case we shall just talk about transducers over Σ and omit the output alphabet in the description of the transducer. The size of a transducer $T = (\Sigma, \Gamma, Q, \Delta, q_\iota, F)$ is its number of transitions: $|T| = |\Delta|$.

We define the following notion of *identity transducer*, which we may use from time to time.

Definition 14. Given a word automaton $\mathcal{A}$, we define the synchronous transducer $T_{\mathcal{A}}$ that accepts the identity relation over $\mathcal{L}(\mathcal{A})$. Formally, if $\mathcal{A} = (\Sigma, Q, \Delta, q_\iota, F)$, we let $T_{\mathcal{A}} = (\Sigma, \Sigma, Q, \Delta', q_\iota, F)$, where $\Delta' = \{(q, a, a, q') \mid (q, a, q') \in \Delta\}$.

We turn to another subclass of rational relations, known as *recognizable relations*.

Definition 15 (Recognizable relations)**.** Let Σ and Γ be two alphabets. A binary relation $\rightsquigarrow \subseteq \Sigma^* \times \Gamma^*$ is *recognizable* if there are two finite families of regular languages $\mathcal{L}_1, \ldots, \mathcal{L}_n \subseteq \Sigma^*$ and $\mathcal{L}'_1, \ldots, \mathcal{L}'_n \subseteq \Gamma^*$ such that $\rightsquigarrow = \bigcup_{i=1}^{n} \mathcal{L}_i \times \mathcal{L}'_i$.

We note Rec the set of recognizable relations. The following inclusions are well-known (Frougny and Sakarovitch, 1993): Rec $\subsetneq$ Reg $\subsetneq$ Rat.

While in general transducers that recognize rational or regular binary relations have to parse the two input words in parallel, a recognizable relation can be recognized by a finite word automaton that starts by reading entirely the first word (or its mirror), and then the second one, before deciding whether these two words are related or not.

The following fact is folklore:

Fact 4. *A relation $\rightsquigarrow \subseteq \Sigma^* \times \Gamma^*$ is recognizable if, and only if, the language $\{\overline{u}\#v \mid u \rightsquigarrow v\}$ is regular, where $\# \notin \Sigma \cup \Gamma$ is a fresh symbol.*

Given a recognizable relation $\rightsquigarrow$, we write $\mathcal{B}_{\rightsquigarrow} = (\Sigma \cup \{\#\}, Q_{\rightsquigarrow}, \delta_{\rightsquigarrow}, s_\iota, F_{\rightsquigarrow})$ for the minimal deterministic word automaton of the language $\{\overline{u}\#v \mid u \rightsquigarrow v\}$ (Fact 4).

This terminates the preliminaries of this thesis, and we now turn to the definition and study of uniform strategies.

3. These definitions are normally theorems, the real definitions are in terms of rational/regular subsets of the monoid $\Sigma^* \times \Gamma^*$. We choose to take these definitions instead as we are not interested in algebraic considerations.

Chapter 3

Uniform strategies

In this chapter we present our notion of uniform strategies. We first introduce the logical language $\mathcal{L}_{\rightsquigarrow}$, which is an extension of CTL^* with two new quantifiers. Roughly speaking, the *strict quantifier* $\boxminus$ universally quantifies over all related plays *in the strategy*, while the *full quantifier* $\boxplus$ universally quantifies over all related plays *in the arena*. We then define the notion of uniform strategies, where uniformity properties are $\mathcal{L}_{\rightsquigarrow}$ formulas. We illustrate this concept by rephrasing various concepts of the literature in terms of uniform strategies. First, we show that the classic imperfect-information strategies can be easily described in our setting. We then describe semantic games for logics of imperfect information, as well as their generalization called *second-order reachability games*. They are different from imperfect-information games, but they still strongly rely on a certain kind of constraints on strategies that involve sets of related plays. We show how we capture this sort of strategies as well. In both these examples we only use the strict quantifier, as these properties on strategies only concern plays in the outcome of strategies. Finally we show with an example that winning strategies for games with epistemic temporal winning condition are easily characterized as uniform strategies. Moreover, the example that we choose illustrates the use of the full quantifier to capture the knowledge of a player who does not know the strategy. It is also a simple example where considering the knowledge of a player who does not know the strategy comes naturally.

3.1 Uniform strategies

In this section we first define the logic $\mathcal{L}_{\rightsquigarrow}$, which can be seen as a generalization of logics of knowledge and branching time, with arbitrary relations between nodes of trees and two different semantics for the "knowledge-like" quantifiers. We then define our notion of uniform strategies, where the uniformity constraints are expressed in this logic.

3.1.1 The logic $\mathcal{L}_{\rightsquigarrow}$

As justified in Section 1.2, we choose to represent uniformity properties of strategies by formulas from a logical language, that we call $\mathcal{L}_{\rightsquigarrow}$. In order to talk about the dynamics of strategy trees (the vertical dimension), $\mathcal{L}_{\rightsquigarrow}$ contains the classic operators of the full branching time logic CTL^*, and as for CTL^* a state formula is interpreted in a node

of a labelled tree. In order to manage the horizontal dimension, the logic also contains quantifiers $\boxminus$ and $\boxbslash$, that universally quantify over "related" nodes of labelled trees (representing paths in a game arena). The semantics of $\mathcal{L}_{\rightsquigarrow}$ is therefore parameterized by a binary relation $\rightsquigarrow$ between nodes of trees.

The difference between the two quantifiers $\boxminus$ and $\boxbslash$ lies in the domain over which they range. Sometimes one wants to quantify only over those related paths that are inside the strategy under consideration. This is the role of $\boxminus$, called the *strict* quantifier. On the other hand it is sometimes natural to consider related paths that are not generated by the strategy, and that do not even start in the initial position of the game. This is achieved by $\boxbslash$, called the *full* quantifier; its semantics is parameterized by a forest of labelled trees called the *universe*, and it quantifies over all $\rightsquigarrow$-related nodes in this universe. We now present the syntax and semantics of $\mathcal{L}_{\rightsquigarrow}$. We recall that AP is some finite set of atomic propositions.

Syntax

The set of well-formed $\mathcal{L}_{\rightsquigarrow}$ formulas is given by the following grammar:

$$\begin{array}{rl} \text{State formulas:} & \varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{A}\psi \mid \boxminus\varphi \mid \boxbslash\varphi \\ \text{Path formulas:} & \psi ::= \varphi \mid \neg\psi \mid \psi \vee \psi \mid \mathbf{X}\psi \mid \psi\mathbf{U}\psi, \end{array}$$

where $p \in AP$. Classically, we define $\texttt{true}$ as $p \vee \neg p$, $\texttt{false}$ as $\neg\texttt{true}$, and we define the Boolean conjunction $\wedge$ for both types of formulas. Also, for each path formula ψ, we write $\mathbf{E}\psi$ for the state formula $\neg\mathbf{A}\neg\psi$, $\mathbf{F}\psi$ for $\texttt{true}\mathbf{U}\psi$, $\mathbf{G}\psi$ for $\neg\mathbf{F}\neg\psi$ and for each state formula φ, we write $\diamondminus\varphi$ for $\neg\boxminus\neg\varphi$ and $\diamondbslash\varphi$ for $\neg\boxbslash\neg\varphi$.

Like for first-order logic, for a formula $\varphi \in \mathcal{L}_{\rightsquigarrow}$, we define its *size* $|\varphi|$ as the number of symbols it contains, and $Sub(\varphi)$ as the set of subformulas of φ.

For example, the intended meaning of the $\mathcal{L}_{\rightsquigarrow}$ state formula $\mathbf{AG}p \to \boxminus p$ evaluated at the root of some labelled tree t, is that at every node x of t that is labelled with the atomic proposition p, it holds that all the nodes *of the tree* related to x are also labelled with p. The meaning of $\mathbf{AG}p \to \boxbslash p$ would be that for every node x of the tree t labelled with p, it holds that all the nodes *in the universe* related to x are labelled with p.

Semantics

Let Υ be a finite set of directions, and let $\Sigma = 2^{AP}$ be the set of possible valuations (for some finite set AP of atomic propositions). As we said, like for CTL^*, an $\mathcal{L}_{\rightsquigarrow}$ state formula (resp. path formula) is interpreted in a node (resp. branch) of a Σ-labelled Υ-tree, but the semantics is parameterized by, first, a binary relation $\rightsquigarrow$ between finite words over Σ, and second, a forest of Σ-labelled Υ-trees which we see as the universe.

Let $\rightsquigarrow$ be a binary relation over Σ^* and $\mathcal{U}$ be a (Σ, Υ)-forest. For two nodes $x, y \in \mathcal{U}$ we let $x \rightsquigarrow y$ denote that $w(x) \rightsquigarrow w(y)$ (that is their node words are related by $\rightsquigarrow$).

Given a (Σ, Υ)-tree $t = (\tau, \ell)$, we define the semantics of $\mathcal{L}_{\rightsquigarrow}$ as follows, where $x \in \tau$ is a node and λ is a branch in τ. $\rightsquigarrow, \mathcal{U}, t, x \models \varphi$ means that the $\mathcal{L}_{\rightsquigarrow}$ state formula φ holds at the node x of the labelled tree t, in the universe $\mathcal{U}$ and with relation $\rightsquigarrow$; similarly, $\rightsquigarrow, \mathcal{U}, t, \lambda \models \psi$ means that the $\mathcal{L}_{\rightsquigarrow}$ path formula ψ holds on the branch λ of t, in universe $\mathcal{U}$ and with relation $\rightsquigarrow$. Since the relation and the universe will always be clear from the

context, we will omit them and simply write $t, x \models \varphi$ and $t, \lambda \models \psi$. The semantics of $\mathcal{L}_{\rightsquigarrow}$ is as follows:

$t, x \models p$ if $p \in \ell(x)$
$t, x \models \neg\varphi$ if $t, x \not\models \varphi$
$t, x \models \varphi_1 \vee \varphi_2$ if $t, x \models \varphi_1$ or $t, x \models \varphi_2$
$t, x \models \mathbf{A}\psi$ if for all $\lambda \in Branches(x)$, $t, \lambda \models \psi$
$t, x \models \boxminus\varphi$ if for all $y \in t$ such that $x \rightsquigarrow y$, $t, y \models \varphi$
$t, x \models \boxminus\!\!\boxminus\varphi$ if for all $y \in \mathcal{U}$ such that $x \rightsquigarrow y$, $\mathcal{U}_y, y \models \varphi$ [1]
$t, \lambda \models \varphi$ if $t, \lambda[0] \models \varphi$
$t, \lambda \models \neg\psi$ if $t, \lambda \not\models \psi$
$t, \lambda \models \psi_1 \vee \psi_2$ if $t, \lambda \models \psi_1$ or $t, \lambda \models \psi_2$
$t, \lambda \models \mathbf{X}\psi$ if $t, \lambda^1 \models \psi$
$t, \lambda \models \psi_1 \mathbf{U} \psi_2$ if there exists $i \geq 0$ such that $t, \lambda^i \models \psi_2$ and
for all $0 \leq j < i$, $t, \lambda^j \models \psi_1$

We shall use the notation $t \models \varphi$ for $t, r \models \varphi$, where r is the root of t. In particular, $t \models \mathbf{A}\psi$ if every branch of t that starts at the root satisfies ψ. Also, for a 2^{AP}-labelled game arena $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$, a position $v \in V$ and a state formula $\varphi \in \mathsf{CTL}^*$, $v \models \varphi$ classically means $Paths_*(v) \models \varphi$.

Remark 4. For convenience, we will sometimes define the relation $\rightsquigarrow$ directly on nodes instead of their labellings, *i.e.* on Υ^* instead of Σ^*. In these cases, we will assume that for each of the finitely many directions $d \in \Upsilon$ there is an atomic proposition $p_d \in AP$, and we will consider trees for which the label of a node $x = d_1 \dots d_n$ contains p_d if, and only if, $d_n = d$. Under this assumption the relation can be equivalently defined on Σ^*.

3.1.2 Uniform strategies

Let $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$ be a finite 2^{AP}-labelled game arena for some finite set AP. Let us note $\Sigma = 2^{AP}$, and let $\rightsquigarrow$ be a binary relation over Σ^*. Finally, let φ be an $\mathcal{L}_{\rightsquigarrow}$ formula. The universe, *i.e.* the range of the full quantifier, is determined by the set V_ι of possible starting positions. We let $\mathcal{U}$ be the (Σ, V)-forest of all paths in the arena starting from a position in V_ι: $\mathcal{U} = Paths_*(V_\iota)$.

Definition 16 (Uniform strategies)**.** A strategy σ is $(\rightsquigarrow, \varphi)$-*uniform* if the strategy tree of σ satisfies φ, *i.e.* $t_\sigma \models \varphi$.

In the rest of the chapter we illustrate our concept by showing how it captures various notions from the literature. In particular it subsumes the two notions that have previously been called "uniform strategies" in different communities: imperfect-information strategies and strategies in the game semantics of Dependence logic.

1. Remember that $\mathcal{U}_y$ is the biggest tree in $\mathcal{U}$ that contains y (see Definition 3).

3.2 Games with imperfect information

We first define in our setting a classic formalism of two-player turn-based games with imperfect information. We show that the standard notion of imperfect-information strategy (or observation-based strategy), and hence the essence of imperfect information, is a quite simple instance of our uniform strategies. We also remark that we can capture these strategies with the same $\mathcal{L}_{\rightsquigarrow}$ formula for any assumptions on the observational power and memory abilities of the player: we just have to choose the adequate binary relation between plays.

We consider the framework of two-player imperfect-information games studied for example in Reif (1984); Chatterjee et al. (2006); Berwanger and Doyen (2008). In these games, Player 1 only partially observes the positions of the game, such that some positions are indistinguishable to her, while Player 2 has perfect information (the asymmetry is due to the fact that we consider all possible outcomes of a strategy, and to the focus being on the existence of strategies for Player 1; this setting is also sometimes referred to as *one and a half* player games with imperfect information). Arenas are directed graphs with *actions* on edges. The game is played in *rounds*: in each round, if the position is a node v, Player 1 chooses an available action a, and Player 2 chooses a next position v' reachable from v through an a-edge.

We reformulate this framework in a way that fits our definition of game arenas by putting Player 1's actions inside the positions. Intuitively, in a position v, Player 1 choosing an action a is simulated by a move to position (v, a), after what Player 2 chooses a position v' reachable from v through a.

Formally, we define an *imperfect-information game arena* as a structure $\mathcal{G}^{imp} = (\mathcal{G}, \mathrm{obs})$, where $\mathcal{G} = (V, E, \{v_\iota\}, v_\iota, \mu)$ is a labelled perfect-information game arena in which the positions of the two players are of a different nature: there is a finite set Act of *actions* such that $V_2 = V_1 \times \mathrm{Act}$. So positions in V_1 are of the form v while positions in V_2 are of the form (v, a). The additional component $\mathrm{obs} : V_1 \to \mathrm{Obs}$ is an *observation function* mapping positions of Player 1 to a nonempty finite set Obs of *observations*. For a position $(v, a) \in V_2$, we let $(v, a).act = a$ denote the action it contains. The players must strictly alternate: $E \subseteq V_1 \times V_2 \cup V_2 \times V_1$. Also, because a move of Player 1 represents her choosing an action and not changing of position, we add the following requirement that $v\, E\, (v', a)$ implies $v = v'$; and because a game starts with Player 1 choosing an action, the initial position v_ι is in V_1. We add the classic requirement that the same actions must be available in indistinguishable positions: for all $v, v' \in V_1$, if $\mathrm{obs}(v) = \mathrm{obs}(v')$ then for all $a \in \mathrm{Act}$, $v\, E\, (v, a)$ if and only if $v'\, E\, (v', a)$. This reflects the assumption that a player is able to distinguish positions with different alternatives of actions.

We assume that AP contains a proposition p_a for each action $a \in \mathrm{Act}$ and a proposition p_o for each observation $o \in \mathrm{Obs}$, and we also assume that it contains a special proposition p_1 that marks positions of Player 1: $\mu(v) = \{p_1, p_{\mathrm{obs}(v)}\}$ for $v \in V_1$ and $\mu(v, a) = \{p_a\}$ for $(v, a) \in V_2$.

We now define the *observational equivalence relation* $\approx$, which relates partial plays that are indistinguishable to Player 1. This relation depends on Player 1's assumed capacities. Following a classic approach (*e.g.* Reif (1984); Chatterjee et al. (2006); Berwanger and Doyen (2008)), we suppose that Player 1 has synchronous perfect recall, meaning that she remembers the whole sequence of observations she makes during a play, and that she

knows how many moves have been made. We also assume that she remembers her own actions. This leads to the following definition: we first extend the observation function to partial plays, by letting

$$\begin{aligned} \mathrm{obs}(v_0(v_0,a_1)v_1\dots(v_{n-1},a_n)) &= \mathrm{obs}(v_0)a_1\mathrm{obs}(v_1)\dots a_n \text{ and} \\ \mathrm{obs}(v_0(v_0,a_1)v_1\dots(v_{n-1},a_n)v_n) &= \mathrm{obs}(v_0)a_1\mathrm{obs}(v_1)\dots a_n\mathrm{obs}(v_n), \end{aligned}$$

and two plays are observationally equivalent if they have the same observation:

for $\rho,\rho' \in Plays_*$, $\rho \approx \rho'$ if $\mathrm{obs}(\rho) = \mathrm{obs}(\rho')$.

Observe that if $\rho \approx \rho'$ and ρ ends in V_1, then so does ρ'.

Definition 17. A strategy σ for Player 1 is *observation-based* if for all partial plays $\rho,\rho' \in \mathrm{Out}(\sigma)$ that end in V_1, if $\rho \approx \rho'$ then $\sigma(\rho).act = \sigma(\rho').act$.

Remark 5. As justified in Remark 2, we are not interested in how strategies are defined on partial plays not in the outcome. For this reason, in this definition we only consider plays that follow the strategy.

Observation-based strategies as uniform strategies

First, let us define the $\mathcal{L}_{\rightsquigarrow}$ formula

$$\texttt{SameAct} := \mathbf{AG}(p_1 \to \bigvee_{a\in\mathrm{Act}} \boxminus\mathbf{EX}p_a),$$

which, if $\rightsquigarrow$ is the observation relation of Player 1, holds of a (deterministic) strategy of Player 1 if whenever it is her turn to play, the action a chosen by the strategy is the same in all related partial plays.

It remains to define the binary relation $\rightsquigarrow$ over 2^{AP} that relates observationally equivalent plays. Notice that for a partial play ρ, the projection of its node word $w(\rho)$ on $2^{AP\setminus\{p_1\}}$ is exactly $\mathrm{obs}(\rho)$. Therefore, two plays are observationally equivalent if, and only if, their node words are identical. Writing Id for the identity relation over 2^{AP}, we obtain the following result.

Proposition 3. *A strategy σ for Player 1 is observation-based if, and only if, it is $(Id, \texttt{SameAct})$-uniform.*

First, observe that formula `SameAct` does not involve any full quantifier, so that the set of possible starting positions V_ι (only used to define the range of full quantifiers) is indifferent here. Now to understand why it is the strict quantifier that we need in `SameAct`, consider Figure 3.1, which represents the (first levels of) the unraveling of some imperfect-information arena with two actions, a and b. Circles represent positions of Player 1, squares positions of Player 2, and each position contains its propositional valuation, except for those propositions representing observations, rather represented with colours: Player 1 confuses positions v_1 and v_2, as well as positions v_3 and v_4. Finally, blue arrows represent the observational equivalence relation $\approx$ (self loops are omitted). Let x_1 (resp. x_2) be the node ending in v_1 (resp. v_2), and notice that $x_1 \approx x_2$.

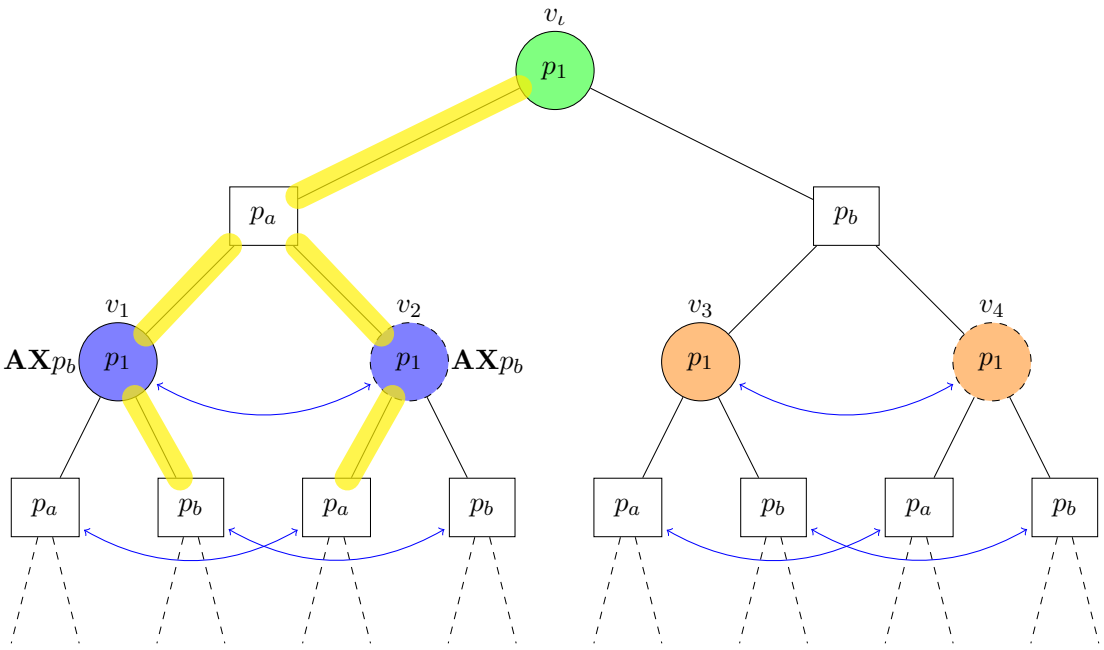

Figure 3.1: An example of strategy in an imperfect-information arena.

The yellow subtree is a strategy tree of Player 1 on which we evaluate `SameAct`. Since this strategy is not observation-based – it plays different actions in x_1 and x_2 – it should not verify `SameAct`. Therefore, because x_1 is a node of the strategy that verifies p_1, for each action $c \in \{a, b\}$, $\boxminus\mathbf{EX}p_c$ should not hold in this node. And indeed the only successor of x_1 *in the strategy tree* is labelled by p_b while the only successor of x_2 *in the strategy tree* is labelled by p_a. If we replaced in `SameAct` the strict quantifier by the full one, according to the semantics of $\boxminus$, nodes v_1 and v_2 would be seen as nodes of the full tree unfolding of the arena, and as such both have two successors, one with p_a and one with p_b, which makes that the formula holds. In fact, by using the full quantifier we would lose track of the strategy under evaluation, and the formula would be true of any strategy.

In this example we assume that Player 1 has perfect recall, and that she can observe her own actions, which determines the definition of $\approx$. It is easy to see that we may capture the same notion of observation-based strategy for any other assumptions on the player's capacities (like asynchronicity, imperfect recall...), by simply replacing the identity relation Id in Proposition 3 with an appropriate one.

3.3 Games for logics of imperfect information

Logics of imperfect information, as they are called, usually come with game semantics that require strategies to be *coherent* in some sense. These constraints capture the semantics of particular features of these logics, and we show that coherent strategies are all instances of our notion of uniform strategies.

3.3.1 Context

Logics of imperfect information start with the introduction of branching quantifiers by Henkin (1961), and continue with the independence-friendly (IF) logic of Hintikka and Sandu (1989). Both introduce into first-order logic some notion of dependence or independence between quantified variables. While it is believed that IF logic does not have a Tarski-style semantics in the classic sense, Hodges (1997) gives a compositional semantics that defines the satisfaction of an IF formula, not by an assignment for free variables, but rather by a *set* of assignments. However, a more intuitive way of defining the semantics of both logics is by means of games with imperfect information. These games are the same as the classic evaluation games for first-order logic, with the only difference that the notions of dependence/independence are very naturally captured by adding imperfect information, hence the name of logics of imperfect information.

Recently, Väänänen started a new trend in logics of imperfect information. In his Dependence Logic (Väänänen, 2007) he proposes, instead of stating dependencies or independencies at the level of quantifiers, to express dependencies directly with a new kind of atomic formulas. These *dependence atoms* are of the form $\mathtt{dep}(x_1, \ldots, x_n, y)$, and mean that variable y only depends on variables $x_1, \ldots, x_n$ or, in other words, that the value of y is completely determined by the values of variables $x_1, \ldots, x_n$. The compositional semantics of Dependence Logic, just like Hodges' semantics for IF logic, is given in terms of sets of assignments that Väänänen calls *teams*. He also gives two game semantics. One is of perfect information but refers explicitly to teams by putting them in positions of the arena, making the game quite different from the classic semantic games for first-order logic or IF logic. The second game semantics, however, is based on the classic arena of these games (see Section 2.4.2), but in order to capture the semantics of his dependence atoms he has to add a constraint on the strategies allowed for Verifier, and he calls this constraint a "uniformity requirement". This constraint makes the game undetermined, reason why Väänänen calls it a game with imperfect information, though it is not clear whether it can be defined in terms of imperfect-information games in the usual acceptation.

Inspired by Dependence Logic, a number of other logics of imperfect information have been defined and studied (Grädel and Väänänen, 2013; Galliani, 2012; Engström, 2012), all consisting in first-order logic with additional atomic formulas with team semantics. They all have semantic games with various uniformity requirements, and in a very recent work Grädel (2013) introduced *second-order reachability games*, which generalize the semantic games for logics with team semantics.

In this section we first describe the example of Dependence logic before turning to the general second-order reachability games.

3.3.2 Dependence Logic

We first give the syntax and the classic compositional team semantics of Dependence Logic. Then we describe how adding to the game semantics of first order logics a particular uniformity constraint on the strategies of Verifier provides a game semantics for Dependence Logic. Finally we show how this constraint on strategies is an instance of uniformity property in our sense.

Syntax and semantics

As already said, Dependence Logic has the same syntax as first-order logic, with additional atomic formulas of the form $\mathtt{dep}(x_1, \ldots, x_n, y)$:

$$\varphi ::= \mathtt{dep}(x_1, \ldots, x_n, y) \mid R(x_1, \ldots, x_n) \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x\varphi$$

Given a model $\mathcal{M} = (D, I)$, we will call *team* a set of assignments. For a team X, a variable x and a function $f : X \to D$, $X[f/x]$ is the team $\{s[f(s)/x] \mid s \in X\}$, and $X[D/x]$ is the team $\{s[a/x] \mid a \in D \text{ and } s \in X\}$.

Dependence Logic is a three valued logic: some formulas may be neither true nor false in a given model and team. For this reason we define two semantics, one for truth ($\models^+$) and one for falsity ($\models^-$). The definition is by mutual recursion:

$\mathcal{M}, X \models^+ \mathtt{dep}(x_1, \ldots, x_n, y)$ if for any pair of assignments $s, s' \in X$ such that $s(x_i) = s'(x_i)$ for all $1 \leq i \leq n$, it holds that $s(y) = s'(y)$

$\mathcal{M}, X \models^- \mathtt{dep}(x_1, \ldots, x_n, y)$ if $X = \emptyset$

$\mathcal{M}, X \models^+ R(x_1, \ldots, x_n)$ if $(s(x_1), \ldots, s(x_n)) \in I(R)$ for all $s \in X$

$\mathcal{M}, X \models^- R(x_1, \ldots, x_n)$ if $(s(x_1), \ldots, s(x_n)) \notin I(R)$ for all $s \in X$

$\mathcal{M}, X \models^+ \neg\varphi$ if $\mathcal{M}, X \models^- \varphi$

$\mathcal{M}, X \models^- \neg\varphi$ if $\mathcal{M}, X \models^+ \varphi$

$\mathcal{M}, X \models^+ \varphi \vee \psi$ if there are two teams Y and Z such that $Y \cup Z = X$, $\mathcal{M}, Y \models^+ \varphi$ and $\mathcal{M}, Z, \models^+ \psi$

$\mathcal{M}, X \models^- \varphi \vee \psi$ if $\mathcal{M}, X \models^- \varphi$ and $\mathcal{M}, X \models^- \psi$

$\mathcal{M}, X \models^+ \exists x\varphi$ if there is $f : X \to D$ such that $\mathcal{M}, X[f/x] \models^+ \varphi$

$\mathcal{M}, X \models^- \exists x\varphi$ if $\mathcal{M}, X[D/x] \models^- \varphi$

So a dependence atom $\mathtt{dep}(x_1, \ldots, x_n, y)$ is true in a team if whenever two assignments agree on the values of the first variables, they must also agree on the value of the last one. This can be seen as reflecting the existence of a functional dependence between the value of y and the values of the other variables. The negation of this definition makes that a dependence atom can only be false[2] in the empty team. But it can also be neither true nor false: for example $\mathtt{dep}(x)$, which means that x is constant, is not true in any team that assigns it at least two different values, but it is not false neither. This three-valued aspect can be alternatively seen as the nondeterminacy of the game semantics we now present.

Semantic game

As we said the semantic game arena is very close to the one for first-order logic described in Section 2.4.2. One notable difference though is that in the case of Dependence Logic, different occurrences of a same subformula must be distinguished. Indeed, $\mathtt{dep}(x) \vee \mathtt{dep}(x)$ is not equivalent to $\mathtt{dep}(x)$, and evaluating whether a strategy verifies the formula requires to know which instance of the atom is hit with which assignments. So we will assume, though we keep it implicit, that syntactic subformulas can be identified, for example with an occurrence number. In addition, in order to "instantiate" the initial team in which the

2. Here "false" refers to falsity ($\models^-$), and not the negation of truth ($\not\models^+$)

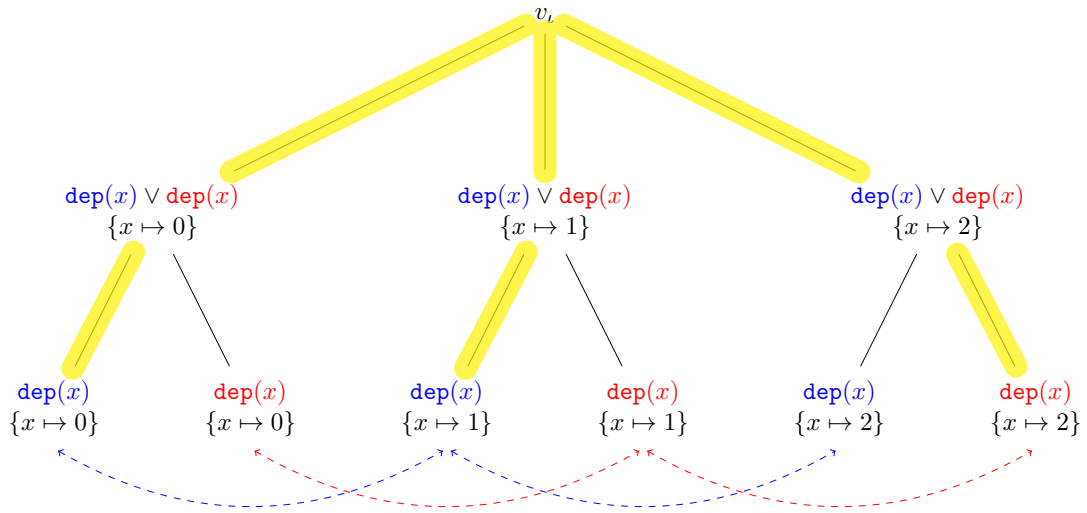

Figure 3.2: Evaluation game for the Dependence Logic formula $\Phi = \mathtt{dep}(x) \vee \mathtt{dep}(x)$ in the team $X = \{\{x \mapsto 0\}, \{x \mapsto 1\}, \{x \mapsto 2\}\}$

formula is evaluated we will add an extra position in which Spoiler starts to play and can choose any of the assignments in the initial team.

Formally, for a Dependence logic formula Φ, a model $\mathcal{M} = (D, I)$ and a team X, we define the game $\mathcal{G}^{\Phi}_{\mathcal{M},X} = (V, E, v_\iota)$, where $V = \mathit{Sub}(\Phi) \times (\mathit{Free}(\Phi) \to D) \times \{\mathit{Ve}, \mathit{Sp}\} \cup \{v_\iota\}$, v_ι belongs to Spoiler, and the moves are as follows. For each $s \in X$, $v_\iota \, E \, (\Phi, s, \mathit{Ve})$, and in a position $(\varphi, s, \mathit{Pl})$ where $\mathit{Pl} \in \{\mathit{Ve}, \mathit{Sp}\}$,

if $\varphi = \mathtt{dep}(x_1, \ldots, x_n, y)$, then Pl wins,
if $\varphi = R(x_1, \ldots, x_n)$, Pl wins if $(s(x_1), \ldots, s(x_n)) \in I(R)$,
if $\varphi = \neg\psi$, the next position is (ψ, s, Pl'), where Pl' is the opponent of Pl,
if $\varphi = \psi_1 \vee \psi_2$, Pl chooses a subformula ψ_i and moves to (ψ_i, s, Pl),
if $\varphi = \exists x \psi$, Pl chooses a value $a \in D$ and moves to $(\psi, s[a/x], \mathit{Pl})$.

Figure 3.2 represents the evaluation game for the formula $\Phi = \mathtt{dep}(x) \vee \mathtt{dep}(x)$ on a model $\mathcal{M}$ with domain $D = \{0, 1, 2\}$, in a team $X = \{\{x \mapsto 0\}, \{x \mapsto 1\}, \{x \mapsto 2\}\}$. The two different syntactic subformulas are distinguished with colours, one is in blue and the other in red. Also, in yellow is depicted a possible strategy for Verifier. In the initial position Spoiler chooses one of the assignments in X, and then Verifier chooses one of the two disjuncts and reaches a terminal position (one that contains an atomic formula). A strategy of Verifier, in this example, represents how she splits the team X into two teams Y and Z, trying to make the disjunction true. Remember that $\mathtt{dep}(x)$ means that x has a constant value, so Verifier must try to split X into two teams, in both of which x is constant. Clearly this is not possible: there is necessarily one team that assigns at least two different values to x, so $\mathcal{M}, X \not\models^{+} \Phi$. For example, the yellow strategy forms the teams $Y = \{\{x \mapsto 0\}, \{x \mapsto 1\}\}$ for the blue dependence atom, and $Z = \{\{x \mapsto 2\}\}$ for the red one, and clearly $\mathcal{M}, Y \not\models^{+} \mathtt{dep}(x)$.

But the fact that a team made of several branches of a strategy verifies or not a dependence atom cannot be captured by classic winning conditions on individual plays. This is the reason why, to capture this semantics, a specific *uniformity requirement* is imposed on strategies for Verifier.

Definition 18. A strategy σ for Verifier is *uniform* in the sense of Väänänen (2007) if, for every two plays $\rho, \rho' \in \mathrm{Out}(\sigma)$ that end in the same (syntactically speaking) atomic dependence subformula, letting $\mathrm{last}(\rho) = (\mathtt{dep}(x_1, \ldots, x_n, y), s, Ve)$ and $\mathrm{last}(\rho') = (\mathtt{dep}(x_1, \ldots, x_n, y), s', Ve)$, if s and s' agree on $x_1, \ldots, x_n$, then they also agree on y.

The following expected property holds:

Proposition 4. *Given Φ a formula of Dependence Logic, $\mathcal{M}$ a model and X a team, $\mathcal{M}, X \models^+ \Phi$ iff Verifier has a winning* uniform *strategy in $\mathcal{G}^{\Phi}_{\mathcal{M},X}$.*

Expressing the uniformity requirement in $\mathcal{L}_{\rightsquigarrow}$

We show how our framework captures this particular notion of uniform strategy, in the case where the domain is finite. Let Φ be a Dependence Logic formula, $\mathcal{M} = (D, I)$ a model and X a team. We decorate $\mathcal{G}^{\Phi}_{\mathcal{M},X}$ with atomic propositions as follows. We assume that we have one atomic proposition $p_a \in AP$ for each value a of the domain, and we mark a terminal position of the form $(\mathtt{dep}(x_1, \ldots, x_n, y), s, Ve)$ with the atomic proposition $p_{s(y)}$. Then we define the binary relation on partial plays $\rightsquigarrow$ as follows: $\rho \rightsquigarrow \rho'$ if ρ and ρ' end in terminal positions with the same dependence atom $\mathtt{dep}(x_1, \ldots, x_n, y)$, and with assignments that agree on $x_1, \ldots, x_n$. On Figure 3.2, dashed arrows represent $\rightsquigarrow$-related plays.

Remark 6. Note that for the sake of clarity we defined $\rightsquigarrow$ on partial plays instead of $(2^{AP})^*$. However, as described in Remark 4, by labelling each position v with an atomic proposition p_v, one can rephrase relation $\rightsquigarrow$ as a binary relation over $(2^{AP})^*$.

Finally we define the following $\mathcal{L}_{\rightsquigarrow}$ formula:

$$\mathtt{AgreeOnLast} := \mathbf{AG} \bigwedge_{a \in D} p_a \to \boxminus p_a,$$

which says that whenever the strategy hits a dependence atom (only these positions can be labeled by some p_a), all related nodes in the strategy tree must be labeled with the same proposition, *i.e.* they must agree on the value a of the last variable. Since related nodes are those that have the same dependence atom and agree on all the first variables, it is obvious that the following holds:

Proposition 5. *A strategy σ for Verifier is uniform in the sense of Väänänen if, and only if, it is $(\rightsquigarrow, \mathtt{AgreeOnLast})$-uniform.*

Once again it is the strict quantifier that must be used here: the team on which a dependence atom is to be evaluated is made of the assignments in terminal positions that contain this dependence atom *and are in the strategy tree*. So the quantification must be restricted to those related plays that are induced by the strategy, which is precisely the semantics of the strict quantifier. In the example, using the full quantifier would make us lose track of the splitting made by the strategy of Verifier at the level of the disjunction.

3.3.3 Second-order reachability games

In the last few years, a number of logics of imperfect information have been studied, which are all extensions of first order logic with additional atomic formulas with second order features. These logics all have team semantics, and more interestingly they also all have game semantics with uniformity constraints on strategies. Examples of these additional atomic formulas are independence atoms (Grädel and Väänänen, 2013), inclusion, exclusion and equiextension atoms (Galliani, 2012), inspired by concepts from database dependency theory. In order to establish general properties of these logics' semantic games, Grädel introduced recently (Grädel, 2013) a notion of games with second-order reachability winning condition. These games capture and generalize the various uniformity constraints, or *consistency criteria* as they are called in Grädel (2013), found in the semantic games for the imperfect-information logics above-mentioned. We present these games and show how in turn our uniform strategies capture these games in the case of finite arenas.

Definition

Second-order reachability games are two-player games played on graphs, that have several particular features. First, they have a set of possible initial positions instead of a unique one, but this can be simulated by letting Player 2 choose the initial position. Therefore, we assume here that they have only one initial position. More interestingly, the graph is acyclic, thus there is a set of terminal positions that have no successor, and all plays are finite; also, in order to capture a generalized version of existential quantification in logics of imperfect information, they consider generalized strategies for Player 1. But the most important particularity is that the winning condition is a set of sets of terminal positions, and a strategy is said to be *consistent winning* if the set of terminal positions it hits is a winning set.

More formally, a second-order reachability game is a tuple $G_{SO} = (V, E, v_\iota, W)$ where (V, E, v_ι) is an acyclic two-player game arena, and letting T be the set of positions without any successor, called *terminal* positions, the winning condition $W \subseteq 2^T$ is a set of winning sets for Player 1. For a generalized strategy σ of Player 1, note $\mathrm{last}(\sigma) = \{\mathrm{last}(\rho) \mid \rho \in \mathrm{Out}(\sigma)\}$ (remember that, in these games, all plays are finite).

Definition 19. A generalized strategy σ for Player 1 is a *consistent winning generalized strategy* if $\mathrm{last}(\sigma) \in W$.

Recasting as uniform strategies

We show that, when considering finite arena, the winning condition of second-order reachability games can be expressed as a uniform property in our framework. First, we transform terminal positions into sink positions, and we mark them with a proposition p_t. Assume also that each position v is labelled with a proposition p_v that identifies it. For a set of positions $S \subseteq V$, we define the formula

$$\nabla_S := \bigwedge_{v \in S} \diamondsuit p_v \wedge \boxminus \bigvee_{v \in S} p_v,$$

which means that the set of related positions in the strategy tree is exactly S (recall that we assume V to be finite here). Now if we let $\rightsquigarrow$ relate all partial plays that end with

terminal positions, and if we define the formula

$$\texttt{ReachWinningSet} := \mathbf{AG}p_t \to \bigvee_{S \in W} \nabla_S,$$

it is clear that:

Proposition 6. *A generalized strategy σ for Verifier is consistent winning if and only if it is $(\rightsquigarrow, \texttt{ReachWinningSet})$-uniform.*

Once again, it is the strict quantifier that must be used here as we want to gather the set of terminal positions *in the strategy*.

For the moment we have given several examples of strategy concepts which rely on some uniformity requirements and can be rephrased in our framework. However, all of them are of a nature that requires using the strict quantifier; the following section shows that some relevant problems can be captured only by the full quantifier.

3.4 Games with opacity condition

We have seen with the previous examples that when the desired property only concerns plays that follow the strategy, the appropriate quantifier to represent it is the strict one. *Games with opacity condition* as studied in Maubert et al. (2011) give an example of a problem that intrinsically requires the full quantifier. They also illustrate that the notion of uniform strategies enables us to express winning conditions with epistemic features.

Games with opacity condition are based on two-player imperfect-information arenas with a particular winning condition, called the *opacity condition*, which involves the knowledge of the player with imperfect information. In such games, some positions are "secret", in the sense that they reveal some critical information. The player with imperfect information (Attacker) aims at obtaining the certainty that the current position is secret, while his opponent (Defender) wants to maintain him under uncertainty.

Assume that a proposition $p_S \in AP$ marks the secret positions. Let $\mathcal{G}^{imp} = (\mathcal{G}, \mathrm{obs})$ be an imperfect-information arena (see Section 3.2), with a distinguished set of positions $S \subseteq V_1$ that denotes the secret. Let $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$ be the arena with $\mu^{-1}(\{p_S\}) = S$ (positions labelled by p_S are exactly positions in S), and let $\approx$ be the observational equivalence relation for Attacker, defined as in Section 3.2. Again, we assume that actions and observations are denoted by atomic propositions.

After a partial play ρ, the *knowledge* or *information set* $I(\rho)$ of Attacker is the set of positions that she considers possible according to her observation of ρ. Formally, for each play $\rho \in Plays_*$ such that $\mathrm{last}(\rho) \in V_1$, we let

$$I(\rho) := \{\mathrm{last}(\rho') \mid \rho' \in Plays_*, \rho \approx \rho'\}.$$

Here it is assumed that Attacker knows the initial position. For this reason, we let $V_\iota = \{v_\iota\}$, and in the definition of information sets, only equivalent *plays* are considered, instead of all equivalent paths in the arena. We could consider instead that Attacker has incomplete information and does not know the exact initial position; to do so we would just define V_ι and information sets differently.

An infinite play π is winning for Defender if Attacker never knows the secret, *i.e.* for all finite prefix ρ of π, not all the positions considered possible by Attacker after ρ are in the secret, *i.e.* $I(\rho) \not\subseteq S$.

We define the formula

$$\texttt{NeverKnowsS} := \mathbf{AG}\neg\boxminus p_S,$$

which says that in all plays, Attacker never knows the secret. It can easily be shown that:

Proposition 7. *A strategy for Defender is winning iff it is* $(Id, \texttt{NeverKnowsS})$*-uniform.*

We give an intuitive explanation of why the full quantifier is the correct one here. According to the definition of Attacker's information sets, she considers that all observationally equivalent plays are possible, even those that are not induced by Defender's strategy. In other words, given a tree representing a strategy for Defender, evaluating whether Attacker knows the secret in some node requires to consider equivalent nodes outside the strategy tree, which is precisely what the full quantifier achieves. Notice that, intuitively, this definition of Attacker's knowledge implies that she "ignores" Defender's strategy. This assumption is natural in this precise setting, and corresponds to the fact that the imperfect information arises from the Attacker not seeing exactly which positions are chosen by Defender.

We describe a slightly different setting, where Attacker still does not see what is the precise current position, but where she knows Defender's strategy. To avoid the information sets to be reduced to singletons we also assume here that Attacker does not know the initial position, but only its observation. Note that in this case we cannot consider Defender's strategies to be only defined on plays starting in the initial position, as this would reveal the initial position to Attacker. Instead, a strategy for Defender must be defined on (at least) all plays that start in any of the positions that Attacker confuses with the initial position. It is then no longer represented by a tree but rather by a forest. We now refine Attacker's information sets with the knowledge of Defender's strategy. Formally, given a strategy σ for Defender, the refined information set I_σ is defined inductively as follows:

$$\begin{aligned} I_\sigma(v_\iota) &= \{v \mid \mathrm{obs}(v) = \mathrm{obs}(v_\iota)\} \text{ and} \\ I_\sigma(\rho \cdot a \cdot v) &= \{\sigma(\rho' \cdot a) \mid \rho' \in I_\sigma(\rho) \text{ and } \mathrm{obs}(v) = \mathrm{obs}(\sigma(\rho' \cdot a))\}. \end{aligned}$$

Informally, at each step, Attacker only considers possible those observationally equivalent plays that may have been induced by Defender's strategy from one of the plays she previously considered possible. In this setting, the appropriate quantifier to model Attacker's knowledge is the strict one.

Note that, here again, we could capture any observational power and memory abilities of Attacker by simply varying the binary relation over plays. Moreover, we consider here a simple example of epistemic temporal objective, which is the formula $\mathbf{AG}\neg\boxminus p_S$, but our framework allows to capture any objective expressible in CTL^* with knowledge.

3.5 Conclusion and related work

In this chapter we defined the logic $\mathcal{L}_{\sim}$ and used it to specify uniformity properties of strategies. Then, we demonstrated the relevance of our notion by capturing all the

motivating examples of strategies with "horizontal" constraints that we described in the introduction of this document. Uniform strategies capture many more problems as, unlike what is usually done in games with imperfect information and epistemic temporal logics, we do not restrict to particular equivalence relations corresponding to precise assumptions on players' abilities. Instead, we allow for arbitrary binary relations between plays, which may not even be equivalence relations.

We have also seen with the last example that the full quantifier $\boxbslash$ can be used to model the knowledge of a player (or an agent) who *does not know the strategy* played by his opponent, while the strict quantifier $\boxminus$ captures the knowledge of a player who *knows* the strategy being played. Up to our knowledge, the question of whether a player knows or not a strategy, though crucial when interpreting knowledge in strategic situations, has surprisingly been discussed only once, in a recent publication (Puchala, 2010).

In this work, Puchala considers two-player turn-based games with imperfect information, where the players have perfect recall and observe their own actions. He first generalizes the classic powerset construction for solving such games with ω-regular winning condition to the setting of *asynchronous* perfect recall, where repetitions of identical observations are deleted. He then studies the problem of deciding the existence of a winning strategy for Player 1 when the winning condition is given in the logic of linear time and knowledge, with knowledge operators for both Player 1 and Player 2. The semantics of the knowledge operators he considers assumes that neither player knows Player 1's strategy. Because it seems unrealistic to assume that a player ignores his own strategy, Puchala makes the following remark. In the setting that he considers, where players have perfect recall and observe their own actions, Player 1 always distinguishes between plays that follow his strategy and plays that do not. Defining Player 1's knowledge independently of his strategy is therefore consistent with the intuition that a player should know her own strategy: the semantics it yields is the same as if his knowledge was restricted to his strategy.

Then, to justify the relevance of the semantics for Player 2's knowledge, Puchala advocates, like we do, that the case where Player 2 knows Player 1's strategy and the case where she ignores it can both be relevant, depending on the situation one wants to model. Puchala gives the example of a network server interacting with a user, and claims that it may be impossible for the user to know the server's protocol.

With this semantics where knowledge does not depend on the chosen strategy, Puchala proves that one can decide whether Player 1 has a winning strategy, both in the synchronous and asynchronous settings. In Chapters 5 and 6 we generalize this result to the case where the indistinguishability relations are arbitrary rational relations.

One may argue that the assumption of Player 2 not knowing Player 1's strategy is not satisfactory as in general, Player 2 may have at least as much computational power as Player 1, and therefore she could make the same strategic calculations and guess what strategy Player 1 will play. However, this supposes that Player 2 knows Player 1's objective, which may not always be the case. But more importantly, in general there is no unicity of a winning strategy, therefore even if Player 2 could compute all winning strategies of Player 1, she could not be sure of which one Player 1 would play. It is therefore not possible in general for Player 2 to know Player 1's strategy, unless Player 1 shares this knowledge.

We have not discussed whether Player 2 knows his own strategy or not. In Puchala's framework, the case is symmetric to that of Player 1, and Player 2 also always can tell the difference between a play that follows his strategy and one that does not, so that the question can be discarded. However, in general, for example where players have imperfect recall and may forget some of their own actions, the matter must be considered. In our framework, we can only either restrict the quantification to Player 1's strategy (with the strict quantifier), or quantify over all related plays in the arena (with the full quantifier). This is a shortcoming that should be addressed in future work, but the right approach to do so is not clear. However, we believe that studying in detail the properties of our strict and full quantifiers may be an interesting first approach to formalize and study the matter of "who knows who does what".

We now start the study of the main problem that we consider in this thesis, which is the synthesis of uniform strategies. We start in the next chapter with uniformity properties expressed in $\mathcal{L}_{\frown}$ but using only the strict quantifier.

Chapter 4

Strictly-uniform strategies

In this chapter, we consider the restriction of $\mathcal{L}_{\rightsquigarrow}$ to formulas that only use strict quantifiers and no full quantifiers, and we call this fragment $\mathcal{SL}_{\rightsquigarrow}$. We investigate the problem of deciding the existence of a strictly-uniform strategy, *i.e.* a uniform strategy for some uniformity property in $\mathcal{SL}_{\rightsquigarrow}$. The input of a decision problem must be finite. Because the binary relations over words used in the semantics of $\mathcal{L}_{\rightsquigarrow}$ are infinite objects in general, we choose to consider the rich class of rational relations, representable by finite state transducers (see Section 2.6). We first show that when the whole class of rational relations is allowed for the semantics of our quantifiers, the existence of a strictly-uniform strategy is undecidable. Then, in an attempt to understand the inherent difficulty of the problem, we introduce the notion of jumping tree automata that generalize alternating tree automata by allowing jumps between nodes of different branches in the input tree. The jumps "implement" the $\boxminus$ quantifiers, and we prove that the satisfiability problem for $\mathcal{SL}_{\rightsquigarrow}$ reduces to the non-emptiness of jumping tree automata. It is then easy to see that deciding the existence of a strictly-uniform strategy also reduces to the non-emptiness of these automata, by taking the product of an automaton that accepts trees representing strategies in the arena with the jumping automaton that accepts models of the formula. We establish that for *recognizable* relations, jumping tree automata can be transformed into equivalent two-way tree automata, whose non-emptiness problem is decidable in Exptime (Vardi, 1998). This yields decidability and a tight 2-Exptime upper bound complexity for the strictly-uniform strategy problem with recognizable relations.

4.1 Undecidability for rational relations

We start with the formal definition of the *strictly-uniform strategy problem* (SUS for short). Let $\mathcal{SL}_{\rightsquigarrow}$ denote the language of *strict uniformity* constraints, *i.e.* the sub-language of $\mathcal{L}_{\rightsquigarrow}$ where the full quantifier ($\boxplus$) is not allowed. The decision problem that we consider is the following. Remark that we use notation Φ instead of φ for the input formula.

Definition 20.

$$\mathsf{SUS} := \left\{ (\mathcal{G}, T, \Phi) \;\middle|\; \begin{array}{c} \mathcal{G} \text{ is a finite } 2^{AP}\text{-labelled arena,} \\ T \text{ is a transducer over } 2^{AP}, \\ \Phi \in \mathcal{SL}_{\rightsquigarrow}, \text{ and} \\ \text{there exists a } ([T], \Phi)\text{-uniform strategy for Player 1 in } \mathcal{G}. \end{array} \right\}$$

It can be proven, by reduction of the Post Correspondence Problem, that SUS is undecidable, but we propose here the stronger result, that we prove in the remaining of the section:

Theorem 7. *The strictly-uniform strategy problem is undecidable for the class of regular equivalence relations.*

Because the class of rational relations contains the class of regular equivalence relations, undecidability of SUS follows.

Corollary 1. SUS *is undecidable.*

We now prove Theorem 7. We reduce the *distributed strategy problem for three-player imperfect-information games with safety objectives*, as addressed by Peterson et al. (2001); Berwanger and Kaiser (2010). We present the problem as stated in Berwanger and Kaiser (2010), in which two players with imperfect information (Player A and Player B) play against nature (the third player). Let Act_A (resp. Act_B) be a finite set of available actions for Player A (resp. Player B), and Obs_A (resp. Obs_B) be a finite set of observations for Player A (resp. Player B). We assume that Act_A and Act_B are disjoint and we write $\mathrm{Act} = \mathrm{Act}_A \times \mathrm{Act}_B$.

A finite *three-player imperfect-information game with safety objective* is a tuple $\mathcal{G}_3^{imp} = (V, E, v_\iota, o_A, o_B)$. V is a finite set of positions with a designated subset $Bad \subseteq V$ of "bad" positions that Player A and Player B should avoid. $E \subseteq V \times \mathrm{Act} \times V$ is a set of transitions and $o_X : V \to \mathrm{Obs}_X$ is an observation function ($X \in \{A, B\}$). In each round, Player X chooses an action $c_X \in \mathrm{Act}_X$, which gives an action profile $x = (c_A, c_B)$, and nature chooses a next position in $E(v, x) = \{v' \mid (v, x, v') \in E\}$. We suppose that all actions are allowed in every position: for all $v \in V$, $a \in \mathrm{Act}_A$ and $b \in \mathrm{Act}_B$, we have $E(v, (a, b)) \neq \emptyset$. The observation functions are extended to partial plays $\rho = v_0 x_0 v_1 \dots x_{n-1} v_n$ by letting $o_X(\rho) = o_X(v_0) o_X(v_1) \dots o_X(v_n)$. Note that actions are not observed.

A strategy for Player X is a partial mapping $\sigma_X : (V \cdot \mathrm{Act})^* \cdot V \to \mathrm{Act}_X$ that assigns an action to each partial play. It must be observation-based: for any partial plays ρ and ρ' such that $o_X(\rho) = o_X(\rho')$, $\sigma_X(\rho) = \sigma_X(\rho')$. A *distributed strategy* is a pair (σ_A, σ_B) of strategies for Player A and Player B. The outcome of a distributed strategy is the set of infinite plays that follow both σ_A and σ_B, and a distributed strategy is winning if no play in the outcome ever visits a position in Bad.

It is well known (Peterson et al., 2001; Berwanger and Kaiser, 2010) that the following problem is undecidable : *given a three-player imperfect-information game with safety objective, does there exist a winning distributed strategy?*

We explain how to reduce it to SUS. We fix an imperfect-information arena $\mathcal{G}_3^{imp} = (V, E, v_\iota, o_A, o_B)$ with observations Obs_A and Obs_B and bad states Bad, and we build a labelled game arena $\mathcal{G} = (V', E', V'_\iota, v'_\iota, \mu)$ in which Player 1 plays for both Player A and

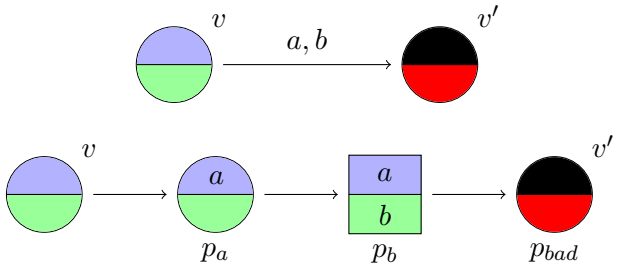

Figure 4.1: Encoding in $\mathcal{G}$ a transition $(v,(a,b),v')$ of $\mathcal{G}_3^{imp}$, with $v' \in Bad$. Colours represent the observations of Player A and Player B.

Player B, and Player 2 plays for nature. Figure 4.1 shows how each transition in $\mathcal{G}_3^{imp}$ is transformed into a widget in $\mathcal{G}$.

The set of positions $V' = V_1^A \uplus V_1^B \uplus V_2$ is split into three: in positions of $V_1^A = V$, Player 1 simulates moves of Player A, in positions of $V_1^B = V \times \mathrm{Act}_A$, Player 1 simulates moves of Player B, and in positions of $V_2 = V \times \mathrm{Act}_A \times \mathrm{Act}_B$, Player 2 simulates moves of nature. Hence for all v, v', a, b, we have $(v,(v,a)) \in E'$, $((v,a),(v,a,b)) \in E'$, and if $(v,(a,b),v') \in E$ then $((v,a,b),v') \in E'$. For each action $c \in \mathrm{Act}_X$, p_c labels positions in which the last move was Player 1 simulating the choice of action c by Player X. In addition, "bad" positions are marked with proposition p_{bad}. Formally, we consider the set of atomic propositions $\{p_c \mid c \in \mathrm{Act}_A \cup \mathrm{Act}_B\} \cup \{p_{bad}\}$, and we label the arena as follows: if $v \in Bad$, $\mu(v) = \{p_{bad}\}$, $\mu(v,a) = \{p_a, p_{bad}\}$ and $\mu(v,a,b) = \{p_b, p_{bad}\}$; otherwise, $\mu(v) = \emptyset$, $\mu(v,a) = \{p_a\}$ and $\mu(v,a,b) = \{p_b\}$. We let $v'_\iota = v_\iota$, and we let $V'_\iota = \{v'_\iota\}$ though it is indifferent here as we will not use the full quantifier $\boxminus$. Finally, the observation functions are defined on this new arena as follows: for a partial play $\rho = v_0(v_0,a_0)(v_0,a_0,b_0)v_1\ldots$, we note $o_X(\rho) = o_X(v_0)o_X(v_1)\ldots$

Clearly, since Player 1 plays for the coalition $\{A,B\}$, we expect each branch of her strategy to satisfy the following path formula:

$$\psi_{Safe} := \mathbf{G}\neg p_{bad}$$

We want to enforce that when Player 1 simulates a move of Player X, her choice is only based on Player X's observation. To do so, we define the symmetric and transitive relation $\rightsquigarrow$ over V'^* that relates two sequences of positions if they end in positions belonging to the same player, and are observationally equivalent for this player:

$$\rightsquigarrow := \left\{ (\rho,\rho') \;\middle|\; \begin{array}{l} \mathrm{last}(\rho) \in V_1^A \text{ and } \mathrm{last}(\rho') \in V_1^A \text{ and } o_A(\rho) = o_A(\rho'), \text{ or} \\ \mathrm{last}(\rho) \in V_1^B \text{ and } \mathrm{last}(\rho') \in V_1^B \text{ and } o_B(\rho) = o_B(\rho') \end{array} \right\}$$

The following path formula states that whenever Player 1 simulates a move of Player X, she chooses the same action in all plays observationally equivalent for Player X:

$$\psi_{Obs} := \mathbf{G} \bigwedge_{c \in \mathrm{Act}_A \cup \mathrm{Act}_B} \mathbf{X}p_c \to \boxminus \mathbf{A}\mathbf{X}p_c$$

We get the following reduction:

Lemma 1. *There is a winning distributed strategy in $\mathcal{G}_3^{imp}$ if, and only if, there is a $(\rightsquigarrow, \mathbf{A}(\psi_{Obs} \wedge \psi_{Safe}))$-uniform strategy for Player 1 in $\mathcal{G}$.*

Proof. A partial play ρ in $\mathcal{G}$ is called *complete* if $\rho \in (V_1^A \cdot V_1^B \cdot V_2)^* \cdot V_1^A$, *i.e.* it corresponds to a sequence of complete rounds in $\mathcal{G}_3^{imp}$. We define a bijection f between partial plays in $\mathcal{G}_3^{imp}$ and complete partial plays in $\mathcal{G}$. For a partial play

$$\rho = v_0(a_0, b_0)v_1 \dots v_{n-1}(a_{n-1}, b_{n-1})v_n$$

in $\mathcal{G}_3^{imp}$, we define

$$f(\rho) = v_0(v_0, a_0)(v_0, a_0, b_0)v_1 \dots v_{n-1}(v_{n-1}, a_{n-1})(v_{n-1}, a_{n-1}, b_{n-1})v_n$$

for the corresponding partial play in $\mathcal{G}$. So for a complete partial play

$$\rho = v_0(v_0, a_0)(v_0, a_0, b_0)v_1 \dots v_{n-1}(v_{n-1}, a_{n-1})(v_{n-1}, a_{n-1}, b_{n-1})v_n$$

in $\mathcal{G}$, we have

$$f^{-1}(\rho) = v_0(a_0, b_0)v_1 \dots v_{n-1}(a_{n-1}, b_{n-1})v_n.$$

We also define a surjective mapping g from the set of all partial plays in $\mathcal{G}$ to partial plays in $\mathcal{G}_3^{imp}$. If ρ is a complete partial play in $\mathcal{G}$, $g(\rho) = f^{-1}(\rho)$. If it is not complete, $g(\rho) = f^{-1}(\rho')$, where ρ' is the longest prefix of ρ that is a complete partial play.

Take a winning distributed strategy (σ_A, σ_B) in $\mathcal{G}_3^{imp}$. We define the strategy σ for Player 1 in $\mathcal{G}$:

$$\sigma(\rho) = \begin{cases} \sigma_A(g(\rho)) & \text{if } \mathrm{last}(\rho) \in V_1^A \\ \sigma_B(g(\rho)) & \text{if } \mathrm{last}(\rho) \in V_1^B \end{cases}$$

First remember that in a tree representing a strategy for Player 1, each node where it is her turn to play has a unique child. Now because σ_A and σ_B are observation-based, it is easy to see that all branches of t_σ verify ψ_{Obs}, and because (σ_A, σ_B) is winning in $\mathcal{G}_3^{imp}$, all branches of t_σ verify ψ_{safe}, hence σ is $(\rightsquigarrow, \mathbf{A}(\psi_{Obs} \wedge \psi_{Safe}))$-uniform.

Now take a $(\rightsquigarrow, \mathbf{A}(\psi_{Obs} \wedge \psi_{Safe})$-uniform strategy σ in $\mathcal{G}$. We define σ_A and σ_B. Take $\rho \in (V \cdot \mathrm{Act})^* \cdot V$ a partial play in $\mathcal{G}_3^{imp}$, and assume that $\mathrm{last}(\rho) = v$, $\sigma(f(\rho)) = (v, a)$ and $\sigma(f(\rho) \cdot (v, a)) = (v, a, b)$. Then we let $\sigma_A(\rho) = a$ and $\sigma_B(\rho) = b$.

Because σ is $(\rightsquigarrow, \mathbf{A}(\psi_{Obs} \wedge \psi_{Safe}))$-uniform, σ_A and σ_B are observation based, and (σ_A, σ_B) is a winning distributed strategy. □

We show that $\rightsquigarrow$ is regular. Consider the synchronous transducer $T_{A,B}$ of Figure 4.2. State q_ι is the initial state (ingoing arrow), q_1^A and q_2^B are final states (doubly circled). Transducer $T_{A,B}$ works as follows: before reading a word w, the transducer guesses whether we are interested in Player A or Player B's observation. In the first case it goes to the left, reads w and writes a word w' observationally equivalent for Player A (remember that actions are not observed). The pair (w, w') is accepted if w (and w') indeed ends in a position where it is Player A's turn to play. The second case is symmetric. Note that $\rightsquigarrow$ is not reflexive – words ending in V_2 are related to no word – but its reflexive closure $\sim$ is also regular (plug in $T_{A,B}$ the synchronous transducer for the identity relation). Lemma 1 would also hold for $\sim$, which concludes the proof of Theorem 7.

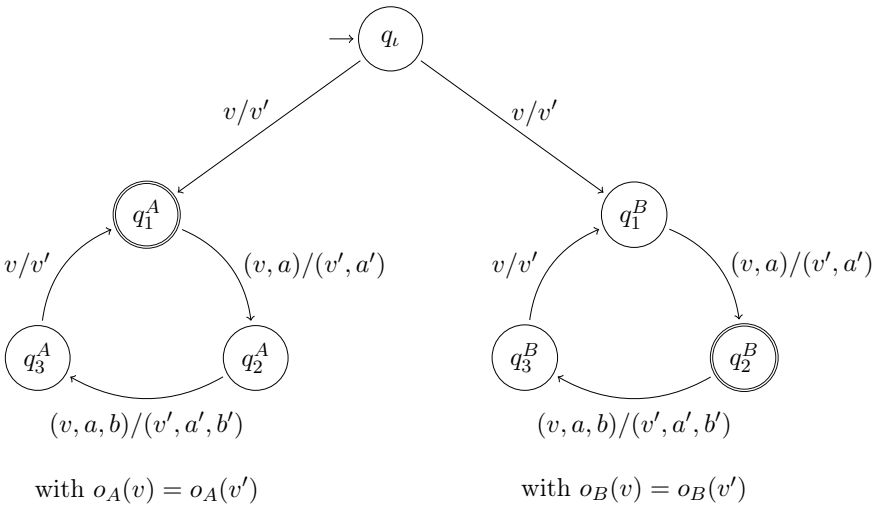

Figure 4.2: The synchronous transducer $T_{A,B}$.

Remark 7. The game $\mathcal{G}$ that we build is a two-player safety game with imperfect information and an unusual indistinguishability relation. Indeed, just like in classic perfect-information two-player safety games, Player 1 must play the same moves in equivalent situations. This is the classic requirement for strategies in imperfect-information games. The only difference with the classic setting is that the indistinguishability relation is not the extension to finite plays of an equivalence relation on positions. Instead, it alternates between two classic perfect-recall synchronous relations, the one of Player A and the one of Player B. This shows that allowing for more general relations than those usually considered in two-player games with imperfect information may quickly lead to undecidability.

We have established the negative but not surprising result that the strictly-uniform strategy problem is undecidable for rational relations. Now, in order to grasp the difficulty of the problem, we define and study an extension of alternating tree automata that captures the semantics of $\mathcal{SL}_{\rightsquigarrow}$.

4.2 Intermezzo: jumping tree automata

Let Υ be a tree alphabet, and let Σ be a labelling alphabet. We define and study *jumping tree automata (JTA)* over Σ-labelled Υ-trees. Jumping tree automata extend classic alternating tree automata with the possibility to send copies of themselves to arbitrary nodes in the input tree, as long as they are related to the current node by some binary relation over Σ^* that equips the automaton.

More precisely, because in general, given a current node of an input tree, there can be infinitely many related nodes, we only allow jumping automata to send a copy to *some* related node or to *all* related nodes. To do so, we introduce two new transition directions, $\diamondsuit$ and $\boxminus$, and we let $Dir_{\rightsquigarrow} = Dir_A \cup \{\diamondsuit, \boxminus\}$ be the set of transition directions for JTA. Recall that Dir_A is the set of transition directions used by alternating tree automata.

Definition 21. A $Dir_{\rightsquigarrow}$-automaton equipped with a binary relation $\rightsquigarrow \subseteq \Sigma^* \times \Sigma^*$ is a *jumping tree automaton (JTA).*[1]

Note that for the moment we do not discuss how the binary relation is represented. Remember also that for convenience we consider that alternating tree automata work with the set of abstract directions $Dir_A = \{\Diamond, \Box\}$ instead of Υ, that $[\Diamond, q]$ sends a copy in state q to some child, and $[\Box, q]$ sends a copy in state q to all children. However, all the results we establish in this chapter also hold in the more general case where concrete directions are allowed.

As in Section 2.5.3, we define the acceptance of a (Σ, Υ)-tree $t = (\tau, \ell)$ in a designated node $x_\iota \in \tau$ by a jumping tree automaton $\mathcal{A}$ as a two-player game between Eve (the proponent) and Adam (the opponent). We repeat the whole definition even though only the rule for jumps is new. Let $t = (\tau, \ell)$ be a (Σ, Υ)-tree, let $x_\iota \in \tau$, and let $\mathcal{A} = (\Sigma, Q, \delta, q_\iota, C)$ be a jumping tree automaton equipped with a relation $\rightsquigarrow \subseteq \Sigma^* \times \Sigma^*$. We define the parity game $\mathcal{G}^{x_\iota}_{\mathcal{A},t} = (V, E, v_\iota, C')$: the set of positions is $V = \tau \times Q \times \mathbb{B}^+(Dir \times Q)$, the initial position is $(x_\iota, q_\iota, \delta(q_\iota, \ell(x_\iota)))$, and a position (x, q, α) belongs to Eve if α is of the form $\alpha_1 \vee \alpha_2$, $[\Diamond, q']$ or $[\diamondsuit, q']$; otherwise it belongs to Adam. Moves in $\mathcal{G}^{x_\iota}_{\mathcal{A},t}$ are defined by the following rules. For clarity we shall abuse notations, and for a node x of t and a state q of $\mathcal{A}$ we write $\delta(q, x)$ for $\delta(q, \ell(x))$. Also, for two nodes $x, y \in \tau$, we write $x \rightsquigarrow y$ for $w(x) \rightsquigarrow w(y)$, *i.e.* the node words of x and y are related by $\rightsquigarrow$.

$$
\begin{array}{ll}
(x, q, \alpha_1 \dagger \alpha_2) \to (x, q, \alpha_i) & \text{where } \dagger \in \{\vee, \wedge\} \text{ and } i \in \{1, 2\}, \\
(x, q, [\bigcirc, q']) \to (y, q', \delta(q', y)) & \text{where } \bigcirc \in \{\Diamond, \Box\} \text{ and } y \text{ is a child of } x, \text{ and} \\
(x, q, [\ominus, q']) \to (y, q', \delta(q', y)) & \text{where } \ominus \in \{\diamondsuit, \boxminus\} \text{ and } x \rightsquigarrow y.
\end{array}
$$

The meaning of the last rule (for the others, see Section 2.5.3) is the following. Because Eve owns positions of the form $(x, q, [\diamondsuit, q'])$ and Adam owns those of the form $(x, q, [\boxminus, q'])$, this rule means that when $\ominus = \diamondsuit$ (resp. $\boxminus$), Eve (resp. Adam) chooses a node related to the current one by $\rightsquigarrow$, and sends in this node a copy of the automaton in state q'. If there is no node y such that $x \rightsquigarrow y$, then $(x, q, [\ominus, q'])$ is a sink position, winning for Adam (resp. Eve) if $\ominus = \diamondsuit$ (resp. $\ominus = \boxminus$). Once again positions of the form $(x, q, \texttt{true})$ and $(x, q, \texttt{false})$ are sink positions, winning for Eve and Adam respectively, and the colouring is inherited from the one of the automaton: $C'(x, q, \alpha) = C(q)$, except for sink positions, which are assigned an even (resp. odd) priority if they are winning for Eve (resp. Adam).

Most of the time the starting node x_ι will be the root r of the tree, and in this case we simply write $\mathcal{G}_{\mathcal{A},t}$ instead of $\mathcal{G}^r_{\mathcal{A},t}$. A tree t is *accepted* by $\mathcal{A}$ if Eve has a winning strategy in $\mathcal{G}_{\mathcal{A},t}$, and we denote by $\mathcal{L}(\mathcal{A})$ the set of trees accepted by $\mathcal{A}$.

We first prove that, just like alternating automata, the class of JTA is closed by complementation. To this aim, for a formula $\alpha \in \mathbb{B}^+(Dir_{\rightsquigarrow} \times Q)$ we define its *dualization* $\widetilde{\alpha}$ by induction as follows: $\widetilde{\texttt{true}} = \texttt{false}$, $\widetilde{\texttt{false}} = \texttt{true}$, $\widetilde{\alpha \vee \beta} = \widetilde{\alpha} \wedge \widetilde{\beta}$, $\widetilde{\alpha \wedge \beta} = \widetilde{\alpha} \vee \widetilde{\beta}$, $\widetilde{[\Diamond, q]} = [\Box, q]$, $\widetilde{[\Box, q]} = [\Diamond, q]$, and, as expected, $\widetilde{[\diamondsuit, q]} = [\boxminus, q]$ and $\widetilde{[\boxminus, q]} = [\diamondsuit, q]$.

Definition 22. Let $\mathcal{A} = (\Sigma, Q, \delta, q_\iota, C)$ be a jumping tree automaton. We define *the complement* of $\mathcal{A}$ by $\widetilde{\mathcal{A}} = (\Sigma, Q, \widetilde{\delta}, q_\iota, \widetilde{C})$, where $\widetilde{C}(q) = C(q) + 1$, and $\widetilde{\delta}(q, a) = \widetilde{\delta(q, a)}$.

1. See Definition 6 for the definition of *Dir*-tree automata.

Lemma 2. *Eve wins* $\mathcal{G}^{x_\iota}_{\widetilde{\mathcal{A}},t}$ *if, and only if, Eve loses* $\mathcal{G}^{x_\iota}_{\mathcal{A},t}$.

Proof. The arenas of both games are isomorphic, and if a position belongs to Eve in $\mathcal{G}^{x_\iota}_{\mathcal{A},t}$ then its counterpart in $\mathcal{G}^{x_\iota}_{\widetilde{\mathcal{A}},t}$ belongs to Adam, and vice versa. Also, a play is winning for a player in one game if and only if its counterpart in the other game is winning for the opponent. From this we have that a winning strategy for a player in one game gives a winning strategy for its opponent in the other, and because parity games are determined, the result follows. □

We now establish that JTA capture $\mathcal{SL}_{\rightsquigarrow}$.

Proposition 8. *Let* φ *be an* $\mathcal{SL}_{\rightsquigarrow}$ *formula, and let* $\rightsquigarrow$ *be a binary relation over* $(2^{AP})^*$. *There exists a jumping tree automaton* $\mathcal{A}_\varphi$ *over alphabet* 2^{AP}, *equipped with* $\rightsquigarrow$, *with two colours and of size* $2^{O(|\varphi|)}$ *such that* $t \in \mathcal{L}(\mathcal{A}_\varphi)$ *if, and only if,* $t \models \varphi$.

Proof. The construction is a simple adaptation of Kupferman et al. (2000), that inductively builds an alternating tree automaton for a CTL^* formula. Notice that because it simplifies the proof, we dually consider as inductive cases $\mathbf{E}\psi$ and $\diamondsuit\varphi$ instead of $\mathbf{A}\psi$ and $\boxdot\varphi$.

$\varphi = p$**:** $\mathcal{A}_\varphi$ has one state q_ι, and $\delta(q_\iota, a) =$ `true` if $p \in a$, `false` otherwise.

$\varphi = \neg\varphi'$**:** $\mathcal{A}_\varphi = \widetilde{\mathcal{A}}_{\varphi'}$.

$\varphi = \varphi_1 \vee \varphi_2$**:** Let $\mathcal{A}_{\varphi_i} = (\Sigma, Q_i, \delta_i, q_i, C_i)$ be the automaton for φ_i, with $Q_1 \cap Q_2 = \emptyset$. Then $\mathcal{A}_\varphi = (\Sigma, \{q_\iota\} \cup Q_1 \cup Q_2, \delta, q_\iota, C)$, where $q_\iota \notin Q_1 \cup Q_2$, $\delta(q, a) = \delta_i(q, a)$ if $q \in Q_i$, $\delta(q_\iota, a) = \delta(q_1, a) \vee \delta(q_2, a)$, $C(q) = C_i(q)$ if $q \in Q_i$ (the colour of q_ι does not matter as in any execution of $\mathcal{A}_\varphi$ q_ι is seen only once).

$\varphi = \mathbf{E}\psi$**, with** ψ **a path formula:** Let $\max(\varphi) = \{\varphi_1, \ldots, \varphi_n\}$ be the set of maximal state sub-formulas of φ. In a first step we see these maximal state sub-formulas as atomic propositions and we work on the alphabet $\Sigma' = 2^{\max(\varphi)}$. Let $\mathcal{A}_\psi = (\Sigma', Q, \Delta, q_\iota, C)$ be a nondeterministic parity word automaton (with two colours) that accepts exactly the models of ψ, where formulas of $\max(\varphi)$ are seen as propositions. This automaton is of size exponential in $|\psi|$ (Vardi and Wolper, 1994). We define the nondeterministic parity tree automaton $\mathcal{A}'_\varphi = (\Sigma', Q, \delta', q_\iota, C)$ that guesses a branch on which it simulates $\mathcal{A}_\psi$. Formally, for $q \in Q$ and $a \in \Sigma'$, we let $\delta'(q, a) = \bigvee_{q' \in \Delta(q,a)} [\diamondsuit, q']$. $\mathcal{A}'_\varphi$ accepts exactly the Σ'-labelled trees that verify φ.
Now from $\mathcal{A}'_\varphi$ we build the automaton $\mathcal{A}_\varphi$ over Σ. This automaton simulates $\mathcal{A}'_\varphi$ and checks that the assumptions made by $\mathcal{A}'_\varphi$ on the truth of formulas in $\max(\varphi)$ are consistent with the input tree. This is achieved by starting additional copies of the automata associated to formulas in $\max(\varphi)$. Formally, let $\mathcal{A}_{\varphi_i} = (\Sigma, Q_i, \delta_i, q_i, C_i)$ be the automaton for φ_i, let $\widetilde{\mathcal{A}}_{\varphi_i} = (\Sigma, \widetilde{Q_i}, \widetilde{\delta_i}, \widetilde{q_i}, \widetilde{C_i})$ be its complement, and assume w.l.o.g that all the state sets are pairwise disjoint. We define the alternating tree automaton $\mathcal{A}_\varphi = (\Sigma, Q \cup \bigcup_i Q_i \cup \widetilde{Q_i}, \delta, C)$, where the colours of states are unchanged, and δ is defined as follows. For states in Q_i (resp. $\widetilde{Q_i}$), δ agrees with δ_i (resp. $\widetilde{\delta_i}$), and finally for $q \in Q$ and $a \in \Sigma$, we let

$$\delta(q, a) = \bigvee_{a' \in \Sigma'} (\delta'(q, a') \wedge \bigwedge_{\varphi_i \in a'} \delta_i(q_i, a) \wedge \bigwedge_{\varphi_i \notin a'} \widetilde{\delta_i}(\widetilde{q_i}, a)).$$

$\varphi = \diamondsuit\varphi'$**, with φ' a state formula:** Let $\mathcal{A}_{\varphi'} = (\Sigma, Q, \delta', q'_\iota, C)$ be the jumping automaton associated to φ'. We simply define $\mathcal{A}_\varphi = (\Sigma, Q \cup \{q_\iota\}, \delta, q_\iota, C)$, where $q_\iota \notin Q$, $\delta(q, a) = \delta'(q, a)$ if $q \in Q$, and $\delta(q_\iota, a) = [\diamondsuit, q'_\iota]$.

A simple proof by induction establishes the size of the automaton and the fact that it works with only two colours. The following lemma states the correction of the construction, which finishes the proof.

Lemma 3. *Take a formula $\varphi \in \mathcal{SL}_\rightsquigarrow$, a labelled tree $t = (\tau, \ell)$ and an initial node $x_\iota \in \tau$. Then Eve wins $\mathcal{G}^{x_\iota}_{\mathcal{A}_\varphi, t}$ iff $t, x_\iota \models \varphi$.*

The proof is by induction on φ. First remind that since the evaluation games of jumping automata are parity games, if a player has a winning strategy then she has a memoryless one (Zielonka, 1998).

$\varphi = p$**:** The only position of $\mathcal{G}^{x_\iota}_{\mathcal{A}_\varphi, t}$ is $(x_\iota, q_\iota, \texttt{true})$ if $p \in \ell(x_\iota)$, $(x_\iota, q_\iota, \texttt{false})$ otherwise. Hence Eve wins if $t, x_\iota \models p$, Adam wins otherwise.

$\varphi = \neg\varphi'$**:** We have $\mathcal{A}_\varphi = \widetilde{\mathcal{A}}_{\varphi'}$. By induction hypothesis, Eve wins $\mathcal{G}^{x_\iota}_{\mathcal{A}_{\varphi'}, t}$ iff $t, x_\iota \models \varphi'$, and by Lemma 2, Eve wins $\mathcal{G}^{x_\iota}_{\widetilde{\mathcal{A}}_{\varphi'}, t}$ iff she loses $\mathcal{G}^{x_\iota}_{\mathcal{A}_{\varphi'}, t}$, hence Eve wins $\mathcal{G}^{x_\iota}_{\mathcal{A}_\varphi, t}$ iff $t, x_\iota \models \neg\varphi'$.

$\varphi = \varphi_1 \vee \varphi_2$**:** The initial position of $\mathcal{G}^{x_\iota}_{\mathcal{A}_\varphi, t}$ belongs to Eve, and she can move either to the initial position of $\mathcal{G}^{x_\iota}_{\mathcal{A}_{\varphi_1}, t}$ or to the one of $\mathcal{G}^{x_\iota}_{\mathcal{A}_{\varphi_2}, t}$, hence Eve wins $\mathcal{G}^{x_\iota}_{\mathcal{A}_\varphi, t}$ iff she wins $\mathcal{G}^{x_\iota}_{\mathcal{A}_{\varphi_1}, t}$ or $\mathcal{G}^{x_\iota}_{\mathcal{A}_{\varphi_2}, t}$. By induction hypothesis, Eve wins $\mathcal{G}^{x_\iota}_{\mathcal{A}_{\varphi_i}, t}$ iff $t, x_\iota \models \varphi_i$, hence she has a winning strategy in $\mathcal{G}^{x_\iota}_{\mathcal{A}_\varphi, t}$ iff $t, x_\iota \models \varphi_1 \vee \varphi_2$.

$\varphi = \mathbf{E}\psi$**:** Suppose that $t, x_\iota \models \mathbf{E}\psi$. There is a path $\pi \in Paths(x_\iota)$ such that $t, \pi \models \psi$. Again, let $\max(\varphi)$ be the set of maximal state sub-formulas of φ, and let $w' \in \Sigma'^\omega$ be the word that agrees with π on the state formulas in $\max(\varphi)$. By definition, $\mathcal{A}_\psi$ has an accepting execution on w'. Now in $\mathcal{G}^{x_\iota}_{\mathcal{A}_\varphi, t}$, Eve can guess the path π, the corresponding word w' and the accepting execution of $\mathcal{A}_\psi$. Assume that in a node x of π, in a state $q \in Q$, Adam challenges Eve on some $\varphi_i \in \max(\varphi)$ that she assumes to be true in x, *i.e.* Adam chooses a transition $\delta_i(q_i, a)$. Note that in the evaluation game this means that Adam moves to position $(x, q, \delta_i(q_i, a))$. Because we have assumed that Eve guesses w' correctly, we have that $t, x \models \varphi_i$. By induction hypothesis, Eve has a winning strategy from the initial position of $\mathcal{G}^x_{\mathcal{A}_i, t}$, which is $(x, q_i, \delta_i(q_i, a))$. Now because $(x, q_i, \delta_i(q_i, a))$ and $(x, q, \delta_i(q_i, a))$ contain the same node x and transition formula $\delta_i(q_i, a)$, a winning strategy in a position is also a winning strategy in the other, and therefore Eve wins Adam's challenge. Idem when Adam challenges her on some φ_i assumed not to be true in a node x: we have that $t, x \not\models \varphi_i$, and by induction hypothesis Eve wins $\mathcal{G}^x_{\widetilde{\mathcal{A}}_{\varphi_i}, t}$.

Now if Eve wins $\mathcal{G}^{x_\iota}_{\mathcal{A}_\varphi, t}$, then there must be a path π in $Paths(x_\iota)$ and a word $w' \in \Sigma'^\omega$ such that w' agrees with π on the formulas in $\max(\varphi)$, and such that there is an accepting run of $\mathcal{A}_\psi$ on w'. Hence $t, \pi \models \psi$, and $t, x_\iota \models \varphi$.

$\varphi = \diamondsuit\varphi'$**:** The initial position of $\mathcal{G}^{x_\iota}_{\mathcal{A}_{\diamondsuit\varphi'}, t}$ belongs to Eve, and the possible successors are the initial positions of the games $\mathcal{G}^y_{\mathcal{A}_{\varphi'}, t}$ with $x \rightsquigarrow y$. Eve thus wins $\mathcal{G}^{x_\iota}_{\mathcal{A}_{\diamondsuit\varphi'}, t}$ iff there is a y such that $x \rightsquigarrow y$ and she wins $\mathcal{G}^y_{\mathcal{A}_{\varphi'}, t}$. The result follows by induction hypothesis.

□

We prove that JTA are adequate machines for the decision problem SUS:

Proposition 9. *Let $(\mathcal{G}, T, \Phi)$ be an instance of* SUS*. There is a jumping tree automaton $\mathcal{A}$ equipped with $[T]$ such that σ is a $([T], \Phi)$-uniform strategy if, and only if, $t_\sigma \in \mathcal{L}(\mathcal{A})$. Moreover, $\mathcal{A}$ can be chosen of size $|\mathcal{A}| = |\mathcal{G}|^2 + 2^{O(|\Phi|)}$ and with only two colours.*

Proof. Let d be the maximum branching degree in $\mathcal{G}$, and let V be the set of positions. Assuming that positions of $\mathcal{G}$ are uniquely identified by atomic propositions, one can build from $\mathcal{G}$ a nondeterministic tree automaton $\mathcal{A}_\mathcal{G}$ (using concrete transition directions $Dir = \{1, \dots, d\}$) that accepts the set of 2^{AP}-labelled $\{1, \dots, d\}$-trees that represent a strategy for Player 1 in $\mathcal{G}$. $\mathcal{A}_\mathcal{G}$ has $|V|$ states and at most $|V| \cdot |\mathcal{G}| \leq |\mathcal{G}|^2$ transitions (see Kupferman et al. (2001), pp. 9-10). Note that this argument works for both deterministic and generalized strategies. Then, by Proposition 8, one can build a JTA $\mathcal{A}_\varphi$ equipped with $[T]$ that accepts the 2^{AP}-labelled $\{1, \dots, d\}$-trees that verify Φ. This automaton is of size $|\mathcal{A}_\Phi| = 2^{O(|\Phi|)}$ and uses two colours. Because of their alternating feature, JTA are trivially closed by language intersection, and therefore one can build in time linear in the sizes of $\mathcal{A}_\mathcal{G}$ and $\mathcal{A}_\Phi$ a JTA that accepts precisely the strategy trees that verify Φ. □

The following is a direct consequence of Theorem 7 and Proposition 9.

Corollary 2. *The emptiness problem for jumping tree automata with regular equivalence relations is undecidable, hence also for rational relations.*

4.3 The special case of recognizable relations

We have seen that SUS is undecidable for regular relations (Theorem 7). We establish that when restricted to recognizable relations the problem becomes decidable and, more precisely, 2-EXPTIME-complete. To achieve this result we prove that JTA equipped with recognizable relations can be effectively transformed into equivalent two-way tree automata of linear size; this implies that the emptiness problem for JTA with recognizable relations is decidable in exponential time. We start with a technical result that allows the transition function of tree automata to test whether the current node is the root of the input tree or not. This is of no interest for one-way automata, but we need this feature in order to simulate JTA with two-way tree automata.

4.3.1 Root-testing two-way tree automata

We slightly extend the definition of two-way tree automata in the following fashion.

Definition 23. A *root-testing two-way alternating parity tree automaton* is a tuple $\mathcal{A} = (\Sigma, Q, \delta, q_\iota, C)$ where Σ, Q, q_ι, C are as for standard two-way parity tree automata, and $\delta : Q \times \Sigma \times \{\texttt{true}, \texttt{false}\} \to \mathbb{B}^+(Dir_\uparrow \times Q)$ is a *root-testing transition function*.

The intended meaning is that when a root-testing two-way tree automaton is in a state q and reads a label a, the transition formula is given by $\delta(q, a, \texttt{true})$ if the current node is the root of the input tree, and $\delta(q, a, \texttt{false})$ otherwise.

The acceptance game for an input Σ-labelled tree $t = (\tau, \ell)$ is defined as in Section 2.5.3, with the following modification of the rules. Let r be the root of t, x be a node of t, and let q be a state of the automaton:

$(x, q, \alpha_1 \dagger \alpha_2) \to (x, q, \alpha_i)$	where $\dagger \in \{\vee, \wedge\}$ and $i \in \{1, 2\}$,
$(x, q, [\bigcirc, q']) \to (y, q', \delta(q', y, \texttt{false}))$	where $\bigcirc \in \{\Diamond, \Box\}$ and y is a child of x [2],
$(x, q, [\epsilon, q']) \to (x, q', \delta(q', x, \texttt{bool}))$	where $\texttt{bool}{=}\texttt{true}$ if $x = r$, $\texttt{false}$ otherwise,
$(x, q, [\uparrow, q']) \to (y, q', \delta(q', y, \texttt{bool}))$	where y is x's parent, and
	$\texttt{bool}{=}\texttt{true}$ if $y = r$, $\texttt{false}$ otherwise.

Note that it is still the case that if x is the root, then the position $(x, q, [\uparrow, q'])$ is a sink position winning for Adam.

We describe how a root-testing two-way automaton can be simulated by a standard two-way tree automaton of linear size.

Proposition 10. *Let $\mathcal{A}$ be a root-testing two-way alternating tree automaton. There is a standard two-way alternating tree automaton $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$ modulo a relabelling of the roots. $\mathcal{A}'$ can be constructed in linear time.*

Proof. Let $\mathcal{A} = (\Sigma, Q, \delta, q_\iota, C)$ be a root-testing two-way alternating tree automaton. Consider a relabelling of Σ-labelled trees over alphabet $\Sigma \cup \Sigma_\#$, where $\Sigma_\#$ is a disjoint copy of Σ (a typical element is $a_\#$ with $a \in \Sigma$) used to label the roots of trees. For a Σ-labelled tree t, let $t_\#$ denote the $\Sigma \cup \Sigma_\#$-labelled tree that is obtained by replacing the label a of t's root with $a_\#$. We say that a $(\Sigma \cup \Sigma_\#)$-labelled tree is *well-formed* if the root, and only the root, is labelled over $\Sigma_\#$. Checking whether a $(\Sigma \cup \Sigma_\#)$-labelled tree is well-formed can be performed by a simple deterministic tree automaton $\mathcal{A}_{wf}$ with three states: the initial state that expects a label in $\Sigma_\#$, another state that expects a label in Σ, and a rejecting state reached if the tree is not well formed. It is then easy to define a standard two-way tree automaton $\mathcal{A}_\#$ over alphabet $\Sigma \cup \Sigma_\#$ that accepts the same language as $\mathcal{A}$, modulo the special labelling of roots. We first define a two-way tree automaton $\mathcal{A}'_\#$ that simulates $\mathcal{A}$ by assuming that the input tree is well-formed: $\mathcal{A}'_\# = (\Sigma \cup \Sigma_\#, Q, \delta_\#, q_\iota, C)$ where $\delta_\#(q, a) = \delta(q, a, \texttt{false})$ for $a \in \Sigma$, and $\delta_\#(q, a_\#) = \delta(q, a, \texttt{true})$ for $a_\# \in \Sigma_\#$. It should be clear that the behaviour of $\mathcal{A}'_\#$ over well-formed $\Sigma \cup \Sigma_\#$-labelled trees simulates the behaviour of $\mathcal{A}$ over Σ-labelled trees. Taking the conjunction of $\mathcal{A}'_\#$ and $\mathcal{A}_{wf}$ yields a two-way alternating tree automaton $\mathcal{A}_\#$ such that $\mathcal{L}(\mathcal{A}_\#) = \{t_\# \mid t \in \mathcal{L}(\mathcal{A})\}$, and the size of $\mathcal{A}_\#$ is linear in the size of $\mathcal{A}$. □

By Proposition 10, all the standard operations on two-way tree automata – especially simulation by nondeterministic automata, Vardi (1998) – can be adapted to root-testing two-way tree automata at no additional cost. In the following we will therefore feel free to let the transition function of two-way tree automata test whether the current node is the root or not.

4.3.2 Simulation of JTA by two-way automata

This section is dedicated to the proof of the following proposition. Remember that if a relation $\rightsquigarrow$ is recognizable, we let $\mathcal{B}_\rightsquigarrow$ be the minimal deterministic word automaton that recognizes it (see Fact 4).

2. y cannot be the root as it is some child of x.

Proposition 11. *If $\mathcal{A}$ is a jumping tree automaton equipped with a recognizable relation $\rightsquigarrow$ and l colours, then there is a two-way tree automaton $\widehat{\mathcal{A}}$ of size $O(|\mathcal{A}| \cdot |\mathcal{B}_{\rightsquigarrow}|)$, with $O(l)$ colours, such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\widehat{\mathcal{A}})$.*

The idea of the proof is as follows: when JTA $\mathcal{A}$ goes down along a branch of a tree, $\widehat{\mathcal{A}}$ behaves likewise. The critical points are the jump instructions of $\mathcal{A}$, say in a node x of the tree. In such a situation, $\widehat{\mathcal{A}}$ stops behaving like $\mathcal{A}$ and enters a *jump mode* that simulates this jump: $\widehat{\mathcal{A}}$ triggers automaton $\mathcal{B}_{\rightsquigarrow}$ and goes up to the root while running $\mathcal{B}_{\rightsquigarrow}$ on the reversed branch. When reaching the root, $\mathcal{B}_{\rightsquigarrow}$ has read $\overline{w(x)}$, the mirror of the node word of x, and $\widehat{\mathcal{A}}$ feeds $\mathcal{B}_{\rightsquigarrow}$ with the # symbol. Then $\widehat{\mathcal{A}}$ goes down along some or all (depending on the jump: respectively existential or universal) branch(es) of the tree while still running $\mathcal{B}_{\rightsquigarrow}$. Each time $\mathcal{B}_{\rightsquigarrow}$ reaches a node y, it has read $\overline{w(x)}\#w(y)$, and if it is currently in a final state then by Fact 4 it means that $x \rightsquigarrow y$; the automaton then (existentially or universally) chooses to continue or to exit the jump mode, in which case $\widehat{\mathcal{A}}$ resumes the simulation of $\mathcal{A}$.

Formally, assume $\mathcal{A} = (\Sigma, Q, \delta, q_\iota, C)$ and $\mathcal{B}_{\rightsquigarrow} = (\Sigma \cup \{\#\}, Q_{\rightsquigarrow}, \delta_{\rightsquigarrow}, s_\iota, F_{\rightsquigarrow})$. Elements of $Q_{\rightsquigarrow}$ are denoted $s, s' \dots$ to avoid confusion with states of the JTA $\mathcal{A}$. We define a two-way automaton $\widehat{\mathcal{A}} = (\Sigma, \widehat{Q}, \widehat{\delta}, \widehat{q}_\iota, \widehat{C})$ that is equivalent to the jumping automaton $\mathcal{A}$ equipped with $\rightsquigarrow$. The set of states is $\widehat{Q} = \widehat{Q}_{\mathcal{A}} \cup \widehat{Q}_{\uparrow} \cup \widehat{Q}_{\downarrow}$, where:

- $\widehat{Q}_{\mathcal{A}} = Q$ is used to simulate $\mathcal{A}$ outside jumps.
- $\widehat{Q}_{\uparrow} = \{\uparrow_{\diamondsuit}, \uparrow_{\boxminus}\} \times Q \times Q_{\rightsquigarrow}$ is used to go up to the root when in jump mode.
- $\widehat{Q}_{\downarrow} = \{\downarrow_{\diamondsuit}, \downarrow_{\boxminus}\} \times Q \times Q_{\rightsquigarrow}$ is used to go down the tree until jump mode ends.

States in $\widehat{Q}_{\uparrow} \cup \widehat{Q}_{\downarrow}$ are used in jump mode. They keep track of three things: first, it is important to identify the existential or universal feature of the jump, e.g. we distinguish from $\uparrow_{\diamondsuit}$ and $\uparrow_{\boxminus}$ in $\widehat{Q}_{\uparrow}$. Second, we must remember the $\mathcal{A}$ state where the standard simulation will resume after the jump mode ends. Third, as the jump mode triggers transitions of $\mathcal{B}_{\rightsquigarrow}$, we must keep track of its current state.

The initial state is $\widehat{q}_\iota = q_\iota$, and we now define the parity condition $\widehat{C}$. Let M denote the maximum colour assigned by C in $\mathcal{A}$. We write M_0 (resp. M_1) for the least even (resp. odd) integer strictly above M [3]. For all $q \in Q$, $s \in Q_{\rightsquigarrow}$, we let $\widehat{C}(q) = C(q)$, $\widehat{C}((\uparrow_{\ominus}, q, s)) = M$, and

$$\widehat{C}((\downarrow_{\ominus}, q, s)) = \begin{cases} M_0 & \text{if } \ominus = \boxminus \\ M_1 & \text{otherwise (i.e. } \ominus = \diamondsuit) \end{cases} \tag{4.1}$$

New colours M_0 and M_1 for states in $\widehat{Q}_{\downarrow}$ are introduced to punish the player in charge of the jump if she happens to never terminate it and remain forever in the jump mode.

Before defining the transition relation, we introduce a mapping $\mathcal{F} : \mathbb{B}^+(Dir_{\rightsquigarrow} \times Q) \to \mathbb{B}^+(Dir_{\uparrow} \times \widehat{Q})$, by induction over the formulas:

- $\mathcal{F}(\texttt{true}) = \texttt{true}$, $\mathcal{F}(\texttt{false}) = \texttt{false}$
- $\mathcal{F}(\alpha \dagger \beta) = \mathcal{F}(\alpha) \dagger \mathcal{F}(\beta)$, for $\dagger \in \{\vee, \wedge\}$
- $\mathcal{F}([\bigcirc, q]) = [\bigcirc, q]$, for $\bigcirc \in \{\Diamond, \Box\}$
- $\mathcal{F}([\ominus, q]) = [\epsilon, (\uparrow_{\ominus}, q, s_\iota)]$, for $\ominus \in \{\diamondsuit, \boxminus\}$

3. Formally, $M_0 = M + 2 - (M \bmod 2)$ and $M_1 = M + 1 + (M \bmod 2)$.

We can now define the transition relation $\widehat{\delta} : \widehat{Q} \times \Sigma \to \mathbb{B}^+(Dir_\uparrow \times \widehat{Q})$, where $\circledcirc \in \{\diamondsuit, \boxminus\}$:

1. $\widehat{\delta}(q, a) = \mathcal{F}(\delta(q, a))$ if $q \in \widehat{Q}_\mathcal{A}$
2. $\widehat{\delta}((\uparrow_\circledcirc, q, s), a) = \begin{cases} [\uparrow, (\uparrow_\circledcirc, q, \delta_\rightsquigarrow(s, a))] & \text{if not at the root,} \\ [\epsilon, (\downarrow_\circledcirc, q, \delta_\rightsquigarrow(s, a \cdot \#))] & \text{otherwise} \end{cases}$
3. $\widehat{\delta}((\downarrow_\circledcirc, q, s), a) = \begin{cases} [\epsilon, q] \dagger [\bigcirc, (\downarrow_\circledcirc, q, \delta_\rightsquigarrow(s, a))] & \text{if } \delta_\rightsquigarrow(s, a) \in F_\rightsquigarrow \\ [\bigcirc, (\downarrow_\circledcirc, q, \delta_\rightsquigarrow(s, a))] & \text{otherwise,} \end{cases}$

   where $(\dagger, \bigcirc) = (\vee, \diamondsuit)$ if $\circledcirc = \diamondsuit$, $(\wedge, \Box)$ otherwise (*i.e.* if $\circledcirc = \boxminus$).

Notice that the size of the two-way tree automaton $\widehat{\mathcal{A}}$ is $|\widehat{\mathcal{A}}| = O(|\mathcal{A}| \times |\mathcal{B}_\rightsquigarrow|)$, and that it uses $l + 2$ colours, where l is the number of colours in $\mathcal{A}$.

Point 1 means that for states in $\widehat{Q}_\mathcal{A} = Q$, the transition function is identical to the one of $\mathcal{A}$, except that an atom of the form $[\circledcirc, q]$ becomes $[\epsilon, (\uparrow_\circledcirc, q, s_\iota)]$, which initiates a jump mode in $\widehat{\mathcal{A}}$. When $\widehat{\mathcal{A}}$ reaches a state $(\uparrow_\circledcirc, q, s_\iota) \in \widehat{Q}_\uparrow$, it initiates the recognizer $\mathcal{B}_\rightsquigarrow$ (the s_ι component), and starts going back to the root, remembering if the jump is existential or universal (the $\uparrow_\circledcirc$ component), and also remembering in which state the simulation of $\mathcal{A}$ shall be resumed when the jump mode terminates (the q component).

Point 2 concerns the case where the current state is $(\uparrow_\circledcirc, q, s) \in \widehat{Q}_\uparrow$, meaning that $\widehat{\mathcal{A}}$ is in the first phase of a jump mode – existential or universal depending on $\circledcirc$. $\widehat{\mathcal{A}}$ is thus going back to the root of the input tree, feeding $\mathcal{B}_\rightsquigarrow$ with the labels it sees on its way. The current state of $\mathcal{B}_\rightsquigarrow$ (before reading the label of the current node) is $s \in Q_\rightsquigarrow$, and $q \in Q$ is the state in which the simulation of $\mathcal{A}$ shall be resumed when the jump mode ends. The transition function tests whether the current node is the root of the tree (see Section 4.3.1). In either case, the transition is deterministic. If the root has not yet been reached, $\mathcal{B}_\rightsquigarrow$ reads the labelling a of the current node, and $\widehat{\mathcal{A}}$ goes to the parent node in state $(\uparrow_\circledcirc, q, \delta_\rightsquigarrow(s, a))$. If the current node is the root, the automaton stays at the root but goes to state $(\downarrow_\circledcirc, q, \delta_\rightsquigarrow(s, a \cdot \#))$. $\downarrow_\circledcirc$ indicates that the automaton enters the second phase of the jump mode. Observe that $\mathcal{B}_\rightsquigarrow$ reads both the label of the root (that has not been read yet) and the special symbol $\#$ that signals the start of the second word (see Fact 4). At this point, one can prove that if the jump mode was initiated in some node x of the input tree, then $\delta_\rightsquigarrow(s, a \cdot \#) = \delta_\rightsquigarrow(s_\iota, \overline{w(x)} \cdot \#)$.

Point 3 concerns the second phase of a jump mode. Assume that the current state of $\widehat{\mathcal{A}}$ is $(\downarrow_\circledcirc, q, s)$, and assume that the current jump mode was initiated in node x of the input tree. In the second phase, if $\downarrow_\circledcirc$ is $\downarrow_\diamondsuit$ (resp. $\downarrow_\boxminus$), then Eve (resp. Adam) chooses step by step a downward path in the tree. The player in charge aims at finding a node y such that $x \rightsquigarrow y$ and where resuming the execution of $\mathcal{A}$ in state q would be winning for her. Let y denote the current node, labelled by a. One can prove that $\delta_\rightsquigarrow(s, a) = \delta_\rightsquigarrow(s_\iota, \overline{w(x)}\#w(y))$. Again there are two cases:

If $\delta_\rightsquigarrow(s, a)$ is an accepting state, then $x \rightsquigarrow y$ by Fact 4. The player in charge (determined by $\circledcirc$) can choose either to terminate the jump mode by choosing atom $[\epsilon, q]$, or to continue looking for a "better" related node by choosing atom $[\bigcirc, (\downarrow_\circledcirc, q, \delta_\rightsquigarrow(s, a))]$. In the first case, the next move is deterministic and resumes normal simulation of $\mathcal{A}$ in state q at node y. In the second case, $\mathcal{B}_\rightsquigarrow$ reads the label a of y, and the player in charge chooses a child of y.

If $\delta_\rightsquigarrow(s, a)$ is not accepting, then the player in charge cannot choose to terminate the

jump mode, but she chooses a child of the current node y to which the automaton goes, and $\mathcal{B}_{\rightsquigarrow}$ reads the label a of y. Observe that if there is no y such that $x \rightsquigarrow y$ then the player in charge of the jump can never terminate it, and Equation (4.1) ensures that she loses. This faithfully simulates the semantics of jumping automata in such cases.

We now show the correctness of $\widehat{\mathcal{A}}$'s definition by comparing the acceptance games of the two automata on some input tree $t = (\tau, \ell)$. Write $\mathcal{G} = \mathcal{G}_{\mathcal{A},t}$ and $\widehat{\mathcal{G}} = \mathcal{G}_{\widehat{\mathcal{A}},t}$ (keeping only reachable positions), and let $\widehat{V}_{\mathcal{A}}$ (resp. $\widehat{V}_{\uparrow}$ and $\widehat{V}_{\downarrow}$) be the set of positions in $\widehat{\mathcal{G}}$ whose state component belongs to $\widehat{Q}_{\mathcal{A}}$ (resp. $\widehat{Q}_{\uparrow}$ and $\widehat{Q}_{\downarrow}$). We consider the bijection $f : V \to \widehat{V}_{\mathcal{A}}$ defined by $f(x, q, \alpha) = (x, q, \mathcal{F}(\alpha))$.

From the above discussion, it is clear that to each move in $\mathcal{G}$ of the form $(x, q, [\ominus, q']) \to (y, q', \alpha)$ can be associated a unique *jump mode* $\widehat{v}_1 \dots \widehat{v}_n$ in $\widehat{\mathcal{G}}$, where $\widehat{v}_1 = f(x, q, [\ominus, q'])$ and $\widehat{v}_n = f(y, q', \alpha)$, which first goes back to the root of the tree before the player in charge of the jump goes down to y and terminates the jump. Similarly, to each finite jump mode in $\widehat{\mathcal{G}}$ corresponds a jump move in $\mathcal{G}$. Concerning the rest of the game, we state the following lemma which follows directly from the definition of $\widehat{\mathcal{A}}$.

Lemma 4. *Let (x, q, α) and (y, q', α') be positions in V. If $\alpha \neq [\ominus, q']$, then*

- *$(x, q, \alpha) \in V$ belongs to Eve in $\mathcal{G}$ iff $f(x, q, \alpha) \in \widehat{V}_{\mathcal{A}}$ belongs to Eve in $\widehat{\mathcal{G}}$.*
- *$(x, q, \alpha) \to (y, q', \alpha')$ iff $f(x, q, \alpha) \mathrel{\widehat{\to}} f(y, q', \alpha')$.*

Finally, Eva has a winning strategy in $\mathcal{G}$ if, and only if, she has one in $\widehat{\mathcal{G}}$, hence $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\widehat{\mathcal{A}})$. This concludes the proof of Proposition 11.

4.3.3 Decidability of SUS for recognizable relations

From Proposition 11, we first get that for each jumping tree automaton equipped with a recognizable relation there is a nondeterministic tree automaton of exponential size that recognizes the same language.

Corollary 3. *Let $\mathcal{A}$ be a jumping tree automaton over alphabet Σ equipped with a recognizable relation $\rightsquigarrow$ and using l colours. Let n_a (resp. n_b) be the number of states in $\mathcal{A}$ (resp. $\mathcal{B}_{\rightsquigarrow}$). Letting $m = n_a \cdot n_b \cdot l$, one can build a nondeterministic tree automaton $\mathcal{A}'$ with $2^{m \log(m)}$ states and $O(m)$ colours such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

Proof. By Proposition 11 there is an equivalent two-way tree automaton $\widehat{\mathcal{A}}$ of size $O(|\mathcal{A}| \cdot |\mathcal{B}_{\rightsquigarrow}|)$ with $O(l)$ colours. More precisely, its number of states is in $O(n_a n_b)$. Therefore, by Theorem 4, $\widehat{\mathcal{A}}$ can be turned into an equivalent nondeterministic tree automaton $\mathcal{A}'$ with $2^{n_a n_b l \log(n_a n_b l)}$ states and $O(n_a n_b l)$ colours. □

It follows from Corollary 3 and Proposition 2 that the nonemptiness problem for jumping automata with recognizable relation is decidable in exponential time.

Corollary 4. *Let $\mathcal{A}$ be a jumping tree automaton over alphabet Σ equipped with a recognizable relation $\rightsquigarrow$ and using l colours. Let n_a (resp. n_b) be the number of states in $\mathcal{A}$ (resp. $\mathcal{B}_{\rightsquigarrow}$), let d be the maximal arity of trees ($d = |\Upsilon|$ for Υ-trees), and let $m = n_a n_b l$. The non-emptiness problem for $\mathcal{A}$ can be decided in time $|\Sigma|^{O(m)} \cdot 2^{O(d\, m^2 \log(m))}$.*

The latter result allows us to solve the strictly-uniform strategy problem for recognizable relations, that we now formally define.

Definition 24.

$$\mathsf{SUS}_{\mathrm{Rec}} := \left\{ (\mathcal{G}, \mathcal{B}_{\rightsquigarrow}, \Phi) \;\middle|\; \begin{array}{c} \mathcal{G} \text{ is a finite } 2^{AP}\text{-labelled arena,} \\ \rightsquigarrow \text{ is a recognizable relation on } (2^{AP})^*, \\ \Phi \in \mathcal{SL}_{\rightsquigarrow}, \text{ and} \\ \text{there exists a } (\rightsquigarrow, \Phi)\text{-uniform strategy for Player 1 in } \mathcal{G}. \end{array} \right\}$$

The size of an instance $(\mathcal{G}, \mathcal{B}_{\rightsquigarrow}, \Phi)$ of $\mathsf{SUS}_{\mathrm{Rec}}$ is the sum of the sizes of its components, plus the number of atomic propositions used: $|(\mathcal{G}, \mathcal{B}_{\rightsquigarrow}, \Phi)| = |\mathcal{G}| + |\mathcal{B}_{\rightsquigarrow}| + |\Phi| + |AP|$.

We establish that $\mathsf{SUS}_{\mathrm{Rec}}$ is 2-Exptime-complete. Note that for the upper bound we could simply invoke Proposition 9 to obtain a jumping automaton of exponential size that accepts tree unfoldings of uniform strategies, and use Corollary 4 to test the emptiness of its language. This gives a decision procedure that is doubly exponential in the size of the formula and exponential in the size of the arena, hence it provides the 2-Exptime upper bound. It is however possible to obtain a decision procedure that is only polynomial in the size of the arena – though still doubly-exponential in the size of the formula. This will be crucial for some results that we establish in Chapter 6.

Proposition 12. *Given an instance* $(\mathcal{G}, \mathcal{B}_{\rightsquigarrow}, \Phi)$ *of* $\mathsf{SUS}_{\mathrm{Rec}}$*, deciding whether* $(\mathcal{G}, \mathcal{B}_{\rightsquigarrow}, \Phi)$ *is a positive instance can be done in time* $(|\mathcal{G}|^d \cdot 2^{|AP|+d\,|\mathcal{B}_{\rightsquigarrow}|\log(|\mathcal{B}_{\rightsquigarrow}|)})^{2^{O(|\Phi|)}}$*, where* d *is the maximum branching degree in* $|\mathcal{G}|$*.*

Proof. Denote by V the set of positions of $\mathcal{G}$, let d be its maximum branching degree, and assume that each position $v \in V$ is identified by an atomic proposition p_v. By Proposition 8, we can build a jumping tree automaton $\mathcal{A}_\varphi$ over alphabet 2^{AP}, equipped with $\rightsquigarrow$, with two colours and $n_a = 2^{O(|\Phi|)}$ states that accepts the $(2^{AP}, \{1, \dots, d\})$-tree models of Φ. Let n_b be the number of states in $\mathcal{B}$. By Corollary 3, we obtain an equivalent nondeterministic tree automaton $\mathcal{A}'_\Phi$ with $2^{O(n_a n_b \log(n_a n_b))}$ states, and $O(n_a)$ colours. Then with the same argument as in the proof of Proposition 9, we build from $\mathcal{G}$ a nondeterministic tree automaton $\mathcal{A}_\mathcal{G}$ with $|V|$ states and $|V| \cdot |\mathcal{G}|$ transitions that accepts the set of strategy trees for Player 1 in $\mathcal{G}$. The acceptance condition of $\mathcal{A}_\mathcal{G}$ is very simple: it has a sink state which is rejecting, and the evaluation game is a safety game where Eve should avoid this rejecting state. It is therefore simple to define the product automaton $\mathcal{A}_{\mathcal{G},\Phi} = \mathcal{A}_\mathcal{G} \times \mathcal{A}'_\Phi$, which is a nondeterministic parity tree automaton with $n = |V| \cdot 2^{O(n_a n_b \log(n_a n_b))}$ states and $l = O(n_a)$ colours. By Proposition 2, testing the non-emptiness of a nondeterministic parity tree automaton over d-ary Σ-labelled trees with n states and l colours can be done in time $\theta = (|\Sigma| \cdot n^{O(d)})^{O(l)}$. Here $\Sigma = 2^{AP}$, and replacing n and l with their expressions we get:

$$\begin{aligned} \theta &= (2^{|AP|} \cdot (|V| \cdot 2^{O(n_a n_b \log(n_a n_b))})^{O(d)})^{O(n_a)} \\ \theta &\leq (2^{|AP|} \cdot |V|^d \cdot 2^{d\, n_a n_b \log(n_a n_b))})^{O(n_a)} \\ &\leq 2^{O(|AP| \cdot n_a)} \cdot |\mathcal{G}|^{O(d \cdot n_a)} \cdot 2^{O(d\, n_a^2 n_b \log(n_a n_b))} \end{aligned}$$

Because $n_a = 2^{O(|\Phi|)}$, we have

$$\begin{aligned} n_a^2 n_b \log(n_a n_b) &= 2^{O(|\Phi|)} \cdot n_b \cdot O(|\Phi|) + 2^{O(|\Phi|)} \cdot n_b \log(n_b) \\ &\leq 2^{O(|\Phi|)} \cdot n_b \cdot 2^{O(|\Phi|)} + 2^{O(|\Phi|)} \cdot n_b \log(n_b) \\ &\leq 2 \cdot 2^{O(|\Phi|)} \cdot n_b \log(n_b) \\ n_a^2 n_b \log(n_a n_b) &\leq 2^{O(|\Phi|)} \cdot n_b \log(n_b) \end{aligned}$$

We finally obtain:

$$\begin{aligned} \theta &\leq 2^{|AP| \cdot 2^{O(|\Phi|)}} \cdot |\mathcal{G}|^{d \cdot 2^{O(|\Phi|)}} \cdot 2^{d \cdot 2^{O(|\Phi|)} \cdot n_b \cdot \log(n_b)} \\ \theta &\leq \left(|\mathcal{G}|^d \cdot 2^{|AP| + d\,|\mathcal{B}_{\rightsquigarrow}| \log(|\mathcal{B}_{\rightsquigarrow}|)}\right)^{2^{O(|\Phi|)}} \end{aligned}$$

□

Remark 8. Observe that Proposition 12 holds also for the variant of $\mathsf{SUS}_{\mathrm{Rec}}$ in which we are interested in generalized uniform strategies. In the proof, the automaton $\mathcal{A}_{\mathcal{G}}$ that accepts strategy trees for Player 1 checks that a node labelled with a position of Player 2 has a child for every possible successor position in the arena; it also checks that a node labelled with a position of Player 1 has exactly one child labelled with a possible next position. For generalized strategies, the automaton just checks the existence of such a child, without enforcing the unicity; this can be implemented with the same number of states and transitions as in the case of deterministic strategies.

We can now state the main theorem of this section.

Theorem 8. $\mathsf{SUS}_{\mathrm{Rec}}$ *is* 2-EXPTIME*-complete.*

Proof. The upper bound comes from Proposition 12, and the lower bound is inherited from the 2-EXPTIME-hardness of solving CTL^* games (Kupferman and Vardi, 1997). □

4.3.4 Synthesis of strictly-uniform strategies

We have seen that the problem $\mathsf{SUS}_{\mathrm{Rec}}$ is decidable. In fact the associated synthesis problem can be solved with the same time complexity. Let $(\mathcal{G}, \mathcal{B}_{\rightsquigarrow}, \Phi)$ be an instance of $\mathsf{SUS}_{\mathrm{Rec}}$, where $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$. We describe how our decision procedure can be adapted to synthesize a finite memory uniform strategy if the answer to the decision problem is positive. Let $\mathcal{A}_{\mathcal{G},\Phi}$ be the nondeterministic parity tree automaton built in the proof of Proposition 12. Recall that $\mathcal{A}_{\mathcal{G},\Phi}$ accepts the set of 2^{AP}-labelled $\{1, \ldots, d\}$-trees – where d is the maximum branching degree in $\mathcal{G}$ – that represent strategies for Player 1 in $\mathcal{G}$ and verify Φ, *i.e.* it accepts the set of tree representations of $(\rightsquigarrow, \Phi)$-uniform strategies. By Theorem 6, if $\mathcal{L}(\mathcal{A}_{\mathcal{G},\Phi}) \neq \emptyset$ – *i.e.* if there exists a uniform strategy – then there is a regular tree in the language, *i.e.* a finitely represented uniform strategy. We describe how this strategy is built. Testing the emptiness of $\mathcal{A}_{\mathcal{G},\Phi}$ is done by building the associated nonemptiness parity game and solving it (see Fact 1). Assume that $\mathcal{L}(\mathcal{A}_{\mathcal{G},\Phi}) \neq \emptyset$. Because parity games are memoryless determined, Player 1 has a memoryless winning strategy σ in the nonemptiness parity game. From such a memoryless winning strategy, Definition 10 describes how one can build a regular tree $t = (\tau, \ell)$ from σ such that $t \in \mathcal{L}(\mathcal{A}_{\mathcal{G},\Phi})$.

Let $\mathcal{T} = (\Upsilon, Q^{\mathcal{T}}, \delta^{\mathcal{T}}, q_\iota^{\mathcal{T}}, F^{\mathcal{T}})$ be the deterministic word automaton of Definition 10 that verifies $\mathcal{L}(\mathcal{T}) = \tau$, and let $\ell_{\mathcal{T}} : Q^{\mathcal{T}} \to 2^{AP}$ be the state labelling such that, for every node $x \in \tau$, $\ell(x) = \ell_{\mathcal{T}}(\delta^{\mathcal{T}}(q_\iota^{\mathcal{T}}, x))$. Remark that $\mathcal{T}$ has the same set of states as $\mathcal{A}_{\mathcal{G},\Phi}$. This regular tree can be adapted to provide a finite memory strategy (Definition 5) for Player 1 in $\mathcal{G}$, and this trategy is uniform. Assume that every position $v \in \mathcal{G}$ is labelled with a proposition $p_v \in AP$. To each state $q \in Q^{\mathcal{T}}$ we assign a unique position $v(q)$ in $\mathcal{G}$: $v(q)$ is the only position $v \in V$ such that $p_v \in \ell_{\mathcal{T}}(q)$. We build the finite memory structure $\mathfrak{M} = (V, M = Q^{\mathcal{T}}, \delta, m_\iota = q_\iota)$ and the mapping $\sigma_{\mathfrak{M}} : M \times V \to V$ that define the uniform strategy. Note that we are in a particular case where each memory state m encodes the current position $v(m)$, so that we can define the mapping simply as $\sigma_{\mathfrak{M}} : M \to V$. The transition function is defined as follows. For $m \in M$, let $v = v(m)$ and note $\{v_1, \ldots, v_i\} = E(v)$ $(i \leq d)$. There are two cases.

- If $v \in V_1$, because t is a (deterministic) strategy tree, there is a unique direction $j \in \{1, \ldots, i\}$ such that $\delta^{\mathcal{T}}(m, j) = m' \in F^{\mathcal{T}}$. We let $\sigma_{\mathfrak{M}}(m) = v(m')$, and $\delta(m, v) = m'$.
- If $v \in V_2$, again because t is a strategy tree, for every $j \in \{1, \ldots, i\}$, we have that $\delta^{\mathcal{T}}(m, j) = m' \in F^{\mathcal{T}}$; we let $\delta(m, v) = m'$, and the definition of $\sigma_{\mathfrak{M}}(m)$ is irrelevant.

One can check that the strategy tree of the finite memory strategy $(\mathfrak{M}, \sigma_{\mathfrak{M}})$ is $t \in \mathcal{L}(\mathcal{A}_{\mathcal{G},\Phi})$, and therefore $(\mathfrak{M}, \sigma_{\mathfrak{M}})$ is a finite memory uniform strategy. Recall that the memory structure has the same number of states as $\mathcal{A}_{\mathcal{G},\Phi}$, which is $|V| \times 2^{O(n_a n_b \log(n_a n_b))}$, where $n_a = 2^{O(|\Phi|)}$ and n_b is the number of states in $\mathcal{B}_{\rightsquigarrow}$. We obtain that a memory doubly exponential in $|\Phi|$ is enough for strictly-uniform strategies with recognizable relations.

4.4 Conclusion and related work

In this chapter we first proved that the existence of uniform strategies for uniformity properties involving only strict quantifiers is undecidable when the relation for the strict quantifier can be an arbitrary relation recognized by a finite state transducer. It remains undecidable even if we restrict to synchronous transducers that recognize equivalence relations. We defined jumping alternating tree automata, that extend alternating tree automata by allowing for jumps between related nodes of the input tree. This feature enables them to capture the semantics of the strict quantifier, and in fact the strictly-uniform strategy problem reduces to the nonemptiness problem for jumping tree automata.

We established that, when restricting to recognizable relations, jumping tree automata can be simulated by classic two-way tree automata, which implies the decidability of their nonemptiness problem. From this we obtained that the strictly-uniform strategy problem is decidable for recognizable relations, and we proved that it is 2-Exptime-complete. Also, relying on classic techniques that, when the language of a tree automaton is not empty, extract a regular tree in the language, we described how a finite memory uniform strategy for Player 1 can be synthesized when there is one.

The fact that solving the strictly-uniform strategy problem for recognizable relations is in 2-Exptime may suggest that for this class of relations, using strict quantifiers in the uniformity properties is "for free". Indeed, solving games with LTL winning condition is already 2-Exptime-complete (Pnueli and Rosner, 1989). We remark however that the complexity of our decision procedure is doubly exponential in the size of the formula and –

for a fixed branching degree – polynomial in the size of the arena, like for LTL games, but in addition, it is also exponential in the size of the automaton that recognizes the relation.

Perfect recall relations are obviously not recognizable, so that our procedure does not allow us in general to synthesize observation-based strategies for players with perfect recall. This is the price to pay for the genericity of the problem considered. However, there are classes of games where the existence of a winning perfect-recall strategy implies the existence of a winning strategy defined on information sets. This is the case for two-player games with ω-regular objectives and (synchronous or asynchronous) perfect recall, and this is the reason why powerset constructions are possible. In such cases, the powerset arena provides an automaton that recognizes the equivalence relation gathering plays of a same information set. Our automata techniques can then be applied.

Also, we believe that the notion of jumping tree automata may be a relevant theoretical tool to study logics with knowledge. For example, it seems that the jumping tree automata may be a good candidate for an automata-theoretic counterpart to the μ-calculus with knowledge. We plan on investigating this question.

It would also be interesting to try to identify a subclass of rational relations that contains synchronous and asynchronous perfect recall relations, and for which the strictly-uniform strategy problem would still be decidable. This may give sufficient conditions on the observational abilities of players for two-player games with imperfect information and perfect recall to be decidable.

We point out that a notion of jumping word automata has recently been proposed by Meduna and Zemek (2012). However their notion is not comparable to ours. First, their automata work on finite words, while ours work on infinite trees. Second, the behaviour of the two sorts of automata are very different. Our automata can progress in an input tree like classic alternating automata do, and in addition they have the possibility to jump inside the tree following some binary relation between nodes. In contrast, jumping automata in Meduna and Zemek (2012) can nondeterministically choose at each transition which letter of the input word is read; the letter is then deleted from the input. Due to this nonstandard behaviour, these automata cannot recognize all regular languages, but they can recognize some context-sensitive languages. For example, they can recognize the language of words that have the same number of each letter.

In the next chapter we consider the case where, instead of strict quantifiers, only full quantifiers are allowed in the uniformity properties.

Chapter 5

Fully-uniform strategies

While in the last chapter we focused on the strict quantifier $\boxminus$, we now turn to the dual case where only the full quantifier $\boxbslash$ is allowed. We consider the decision problem of the existence of a *fully-uniform strategy*, but like in the decidable case for strictly-uniform strategies, our decision procedures can be adapted with no additional cost to effectively provide a uniform strategy whenever there exists one.

We establish that the existence of a fully-uniform strategy is decidable for the class of rational relations, and that it is nonelementary. In order to separate different aspects of the problem, we introduce *information set automata*. They can be seen as a sort of partial determinization of finite state transducers, that recognize information sets for any rational relation between plays of a game. Using these automata, we describe a construction reminiscent of the classic powerset construction for games with imperfect information (Reif, 1984). This construction enables us to evaluate and eliminate the innermost $\boxbslash$ quantifiers in the uniformity property. In general, this procedure has to be repeated a number of time matching the maximum nesting of quantifiers in the formula. This yields a decision procedure that runs in k-Exptime, where k is the maximum nesting of $\boxbslash$ quantifiers in the uniformity property – 2-Exptime for $k \leq 2$. We prove that this decision procedure is essentially optimal by providing the matching lower bounds.

Then we consider a subclass of rational relations, called K45NM, which consists of rational relations that verify transitivity, Euclideanity and *No Miracles* (Halpern and Vardi, 1989; Pacuit and van Benthem, 2006). For this class of relations, we prove that the complexity of the fully-uniform strategy problem collapses to 2-Exptime-complete, which is essentially the same complexity as solving LTL games. In establishing this result we again take advantage of information set automata. We prove that when a relation is in K45NM, its associated information set automaton has the particularity that two related plays take the automaton to *information set bisimilar* states. This allows us to evaluate and eliminate all the $\boxbslash$ quantifiers with a single powerset construction.

The chapter is organized as follows. We first formally define the problems that we consider and we state our results. In Section 5.2 we define information set automata and study their properties. Then we use these automata to describe our decision procedures and establish the upper bounds of our results. We finish the chapter by establishing the matching lower bounds.

5.1 Main results

First, let $\mathcal{FL}_{\rightsquigarrow}$ denote the language of *full uniformity* constraints, *i.e.* the sublanguage of $\mathcal{L}_{\rightsquigarrow}$ that does not allow strict quantifiers ($\boxminus$) but only full ones ($\boxplus$).

The $\boxplus$*-depth* or *nesting depth* of a $\mathcal{FL}_{\rightsquigarrow}$ formula φ, written $d(\varphi)$, is the maximum number of nested $\boxplus$ quantifiers in φ, defined inductively as follows:

$$\begin{array}{lll} d(p) = 0 & d(\neg\varphi) = d(\varphi) & d(\varphi \vee \varphi') = \max(d(\varphi), d(\varphi')) \\ d(\mathbf{A}\psi) = d(\psi) & d(\mathbf{X}\psi) = d(\psi) & d(\psi \mathbf{U} \psi') = \max(d(\psi), d(\psi')) \\ & d(\boxplus\varphi) = 1 + d(\varphi) & \end{array}$$

For $k \in \mathbb{N}$, we let $\mathcal{FL}^k_{\rightsquigarrow} := \{\varphi \in \mathcal{FL}_{\rightsquigarrow} \mid d(\varphi) \leq k\}$.

Definition 25. For each $k \in \mathbb{N}$, we let

$$\mathsf{FUS}_k := \left\{ (\mathcal{G}, T, \Phi) \;\middle|\; \begin{array}{c} \mathcal{G} \text{ is a finite } 2^{AP}\text{-labelled arena,} \\ T \text{ is a transducer over } 2^{AP}, \\ \Phi \in \mathcal{FL}_{\rightsquigarrow}, d(\Phi) \leq k, \text{ and} \\ \text{there exists a } ([T], \Phi)\text{-uniform strategy for Player 1 in } \mathcal{G}. \end{array} \right\}$$

The fully-uniform strategy problem is formally defined as follows:

Definition 26. The *fully-uniform strategy problem* is

$$\mathsf{FUS} := \bigcup_{k \in \mathbb{N}} \mathsf{FUS}_k.$$

We let the size of an instance be the sum of the sizes of its components, plus the number of atomic propositions: $|(\mathcal{G}, T, \Phi)| = |\mathcal{G}| + |T| + |\Phi| + |AP|$.

Theorem 9. FUS_k *is* k-EXPTIME*-complete if* $k \geq 2$*, otherwise it is* 2-EXPTIME*-complete.*

Corollary 5. *The fully-uniform strategy problem is nonelementary.*

Before proving this result, we state the second main result of this chapter.

First, remind that K45 is the set of transitive and Euclidean relations [1], which are "almost" equivalence relations, and are relevant for example in the context of belief revision and modelling plausibility (Fagin et al., 1995).

We define in our setting the classic notion of *No Miracles*:

Definition 27. A binary relation $\rightsquigarrow \subseteq \Sigma^* \times \Sigma^*$ satisfies the *No Miracles* property if for all $u, v, w \in \Sigma^*$, $u \rightsquigarrow v$ implies $u \cdot w \rightsquigarrow v \cdot w$.

We now discuss this definition. The intuitive meaning of the No Miracles property, as defined *e.g.* in Pacuit and van Benthem (2006), is that when two situations are indistinguishable, they should remain indistinguishable if the same events occur in both of them. While the No Miracles property is usually defined with respect to a system (roughly a set of finite histories and a relation between them), here we just consider binary relations

1. We recall that a relation $\rightsquigarrow$ is *transitive* if $u \rightsquigarrow v$ and $v \rightsquigarrow w$ implies $u \rightsquigarrow w$, and it is *Euclidean* if $u \rightsquigarrow v$ and $u \rightsquigarrow w$ implies $v \rightsquigarrow w$.

over some alphabet. Our motivation is that often, the relation representing the observational abilities of an agent, or a player, may be defined independently from the dynamics of the precise arena considered, only with regards to the set of possible positions, or observations. In other words, the relation may often be defined given only the alphabet, without restricting to some language over it that would represent the set of possible plays or histories.

Consider for example the case of synchronous perfect recall. If the set of possible positions V and the observation function $\mathrm{obs} : V \to \mathrm{Obs}$, as defined in Section 3.2, are given, then one can define a transducer that recognizes the synchronous perfect recall relation over finite sequences of positions: two words in V^* are related if they form the same sequences of observations.

We will see that considering the properties of a relation indepently from the arena is sometimes relevant: we will establish that the complexity of the fully-uniform strategy problem is better when the relation defined by the input transducer on Σ^* verifies the No Miracles property, even if its restriction to the set of possible plays in the input arena does not verify it.

Remark 9. We want to remark that the notion of No Miracles as defined for example in Pacuit and van Benthem (2006) defers from the notion of No Learning considered in Halpern and Vardi (1989): while the former says that when two indistinguishable runs of a system are continued *with the same events*, they remain indistinguishable, the latter says roughly that when two indistinguishable runs of a system are continued, no matter how they are continued, they must somehow remain indistinguishable. A system where different events with different observations can occur in indistinguishable situations will never verify the latter notion, while it can verify the former.

We note K45NM for the set of rational relations that verify transitivity, Euclideanity and No Miracles. $\mathsf{FUS}_{\mathrm{K45NM}}$ is the restriction of FUS to K45NM relations.

Definition 28.

$$\mathsf{FUS}_{\mathrm{K45NM}} := \left\{ (\mathcal{G}, T, \Phi) \;\middle|\; \begin{array}{c} \mathcal{G} \text{ is a finite } 2^{AP}\text{-labelled arena,} \\ T \text{ is a transducer over } 2^{AP} \text{ such that } [T] \in \mathrm{K45NM}, \\ \Phi \in \mathcal{FL}_{\sim}, \text{ and} \\ \text{there exists a } ([T], \Phi)\text{-uniform strategy for Player 1 in } \mathcal{G}. \end{array} \right\}$$

Theorem 10. $\mathsf{FUS}_{\mathrm{K45NM}}$ *is* 2-Exptime*-complete.*

This result is of interest as most relations used to represent uncertainty fall into this class, like synchronous or asynchronous perfect-recall relations, and in general all equivalence relations induced by alphabetic morphisms. Indeed, notice that here we see relations as being defined with regards to a particular set of positions (or abstractions thereof), but without considering the dynamics of any particular game arena. Seen as such, the classic synchronous and asynchronous perfect recall relations verify our No Miracles property.

We now introduce information set automata, and then we use them to prove the upper bounds of Theorem 9 and Theorem 10. We prove the lower bounds at the end of the chapter.

5.2 Information set automaton

The notion of information set is classic in games with imperfect information. After a partial play, a player's information set is the set of positions considered possible by this player according to what she has observed so far. We introduce a general notion of information set: informally, given a binary relation over finite words, the information set of a word is the set of all last letters of related words. We then describe how, when given a transducer that recognizes a binary relation, one can build a deterministic word automaton that computes the information set of its input word. We also introduce a bisimulation relation on states of this automaton, called the *information set bisimulation*, and we consider the quotient automaton. We prove that for relations in K45NM, two related plays take the quotient automaton to the same state. This result is crucial for the upper bound of Theorem 10.

Definition 29. Let Σ be an alphabet, and let $\rightsquigarrow \subseteq \Sigma^* \times \Sigma^*$ be a binary relation over Σ^*. For $w \in \Sigma^*$, the *information set* for $\rightsquigarrow$ after the word w is:

$$I_{\rightsquigarrow}(w) = \{a \in \Sigma \mid w \rightsquigarrow w' \cdot a \text{ for some } w' \in \Sigma^*\}$$

For the rest of Section 5.2 we fix a transducer $T = (\Sigma, Q, \Delta, Q_\iota, F)$ over some alphabet Σ, and we note $\rightsquigarrow = [T]$. We describe a powerset construction that transforms T into a deterministic automaton that computes information sets for $\rightsquigarrow$. Transducers cannot be determinized in general, but because we only aim at computing information sets we can afford to forget the output tape, except for the last letter. This allows us to obtain a deterministic automaton $\mathcal{A}^T$, that we call the *information set automaton*.

Computing information sets in the classic framework of extensive imperfect-information games (see Section 3.2) is simple: the player only needs to remember the current information set, and update it with each newly observed position. In the case of rational relations in general, information sets cannot be inferred from the new letter and the previous information set only, and we have to store more information in states of the automaton.

While reading a word w, we remember two things: first, the set of states q that the transducer may have reached by nondeterministic runs on input w, and second, for each such state q, the set $Last(q)$ of all last letters of output words in a run on input w that ends in q. Therefore, states of $\mathcal{A}^T$ are pairs of the form $(S, Last)$, with $S \subseteq Q$ and $Last : Q \to 2^\Sigma$ (recall that Q is the transducer's set of states).

Definition 30 (Information set automaton)**.** The deterministic *information set automaton* for $T = (\Sigma, Q, \Delta, q_\iota, F)$ is $\mathcal{A}^T = (\Sigma, A, \delta, \alpha_\iota)$, where

- $A = 2^Q \times (Q \to 2^\Sigma)$ is the set of states,
- $\alpha_\iota = (S_\iota, Last_\iota)$ is the initial state, with
  - $S_\iota = \{q \mid \exists w \in \Sigma^*,\ q_\iota \xrightarrow{[\epsilon/w]} q\}$ and
  - $Last_\iota(q) = \{a \mid \exists w \in \Sigma^*, q_\iota \xrightarrow{[\epsilon/w \cdot a]} q\}$
- $\delta((S, Last), a) = (S', Last')$, with
  - $S' = \{q' \mid \exists q \in S, \exists w \in \Sigma^*,\ q \xrightarrow{[a/w]} q'\}$ and
  - $Last'(q') = \{a' \mid \exists q \in S,\ \exists w \in \Sigma^*, q \xrightarrow{[a/w \cdot a']} q'$, or $q \xrightarrow{[a/\epsilon]} q'$ and $a' \in Last(q)\}$

First, observe that we do not specify accepting states. This is because $\mathcal{A}^T$ is not meant to recognize a set of words, but rather to compute the information set of any possible input. Second, observe the initial state $(S_\iota, Last_\iota)$: S_ι is the set of states that can be reached internally (*i.e.* by reading nothing) from q_ι, and for each state $q \in S_\iota$, $Last_\iota(q)$ is the set of letters found at the end of the output tape in runs that internally reach q.

We now detail how for a current state $(S, Last)$ and a new input letter a, we build the successor state $\delta((S, Last), a) = (S', Last')$. First, S' is made of all the states q' that can be reached from a state $q \in S$ with a transition of the form $q \xrightarrow{[a/w]} q'$ with $w \in \Sigma^*$. If $w = w' \cdot b$, then b is added to $Last'(q')$, otherwise $w = \epsilon$ and each $b \in Last(q)$ can still be the last letter on the output tape after reading a and reaching q', therefore for each $b \in Last(q)$ we let $b \in Last'(q')$. Notice that by definition $Last'(q') = \emptyset$ for each $q' \notin S'$.

Since $\mathcal{A}^T$ is complete, $\delta(\alpha_\iota, w)$ is defined (in the classic way) for all $w \in \Sigma^*$. We define the size of an information set automaton as its number of transitions: $|\mathcal{A}^T| = |\delta|$. Notice that because $\mathcal{A}^T$ is complete and deterministic, $|\delta| = |A| \cdot |\Sigma|$.

Lemma 5. *$\mathcal{A}^T$ is of size $2^{O(|T|\cdot|\Sigma|)}$ and can be computed in time $2^{O(|T|\cdot|\Sigma|)}$.*

Proof. The number of states of $\mathcal{A}^T$ is $|A| = |2^Q \times (Q \to 2^\Sigma)| = 2^{|Q|(1+|\Sigma|)} = 2^{O(|T|\cdot|\Sigma|)}$, and because $|\mathcal{A}^T| = |\Sigma| \cdot |A|$ we have that $|\mathcal{A}^T| = 2^{O(|T|\cdot|\Sigma|)}$.

Now for the time complexity: for each state $\alpha = (S, Last)$ and each letter $a \in \Sigma$, one has to compute $\delta(\alpha, a) = (S', Last')$. S' and $Last'$ can be computed in quadratic time in the size of Δ. To do so, for each q in S, one computes $S_{q,a} = \{(q', a') \mid \exists w \in \Sigma^*, q \xrightarrow{[a/w\cdot a']} q',$ or $q \xrightarrow{[a/\epsilon]} q'$ and $a' \in Last(q)\}$. S' and $Last'$ can be easily reconstructed from $\cup_{q\in S} S_{q,a}$. For $q \in S$, computing $S_{q,a}$ can be done by depth-first search in the tree unfolding of the transducer, by first reading a and then a series of ϵ. A branch of the search is stopped when there is no more ϵ to read or a state is revisited. So computing $S_{q,a}$ takes time $O(|\Delta|)$. Computing $\delta(\alpha, a)$ requires to do this for each $q \in S$, so it takes time $O(|Q| \cdot |\Delta|) = O(|T|^2)$. Doing so for each $\alpha \in A$ and $a \in \Sigma$ yields a time complexity in $O(|A| \cdot |\Sigma| \cdot |T|^2) = O(|\mathcal{A}^T| \cdot |T|^2) = 2^{O(|T|\cdot|\Sigma|)}$. □

Finally, for every state $\alpha = (S, Last)$ of $\mathcal{A}^T$, we define:

$$\alpha.I := \bigcup_{q \in S \cap F} Last(q).$$

We prove that the components S and $Last$ actually capture what they are meant to:

Lemma 6. *Let $w \in \Sigma^*$. If $(S, Last) = \delta(\alpha_\iota, w)$, then:*

1. *$S = \{q \mid \exists w' \in \Sigma^*, q_\iota \xrightarrow{[w/w']} q\}$, and*
2. *for each $q \in S$, $Last(q) = \{a' \mid \exists w' \in \Sigma^*, q_\iota \xrightarrow{[w/w'\cdot a']} q\}$.*

Proof. The proof is done by induction on w.

Case $w = \epsilon$**.** We have $\delta(\alpha_\iota, \epsilon) = \alpha_\iota = (S_\iota, Last_\iota)$. The result is the mere definition of S_ι and $Last_\iota$.

Case $w = w_1 \cdot a$**.** Letting $(S_1, Last_1) = \delta(\alpha_\iota, w_1)$, we have: $(S, Last) = \delta((S_1, Last_1), a)$. For the left-to-right inclusions of Points 1 and 2, let $q \in S$ and $a' \in Last(q)$. By definition of δ, there is a $q_1 \in S_1$ and a $w_1' \in \Sigma^*$ such that $q_1 \xrightarrow{[a/w_1']} q$, and either

$w'_1 = w'_2 \cdot a'$ for some w'_2, or $w'_1 = \epsilon$ and $a' \in Last_1(q_1)$. By induction hypothesis on w_1, we have that $S_1 = \{q \mid \exists w' \in \Sigma^*, q_\iota \xrightarrow{[w_1/w']} q\}$. Since $q_1 \in S_1$ there exists $w' \in \Sigma^*$ such that $q_\iota \xrightarrow{[w_1/w']} q_1$, and by transitivity, $q_\iota \xrightarrow{[w_1 \cdot a/w' \cdot w'_1]} q$. This proves the left-to-right inclusion of Point 1. For the left-to-right inclusion of Point 2 we split into two cases for w'_1.

- If $w'_1 = w'_2 \cdot a'$ for some w'_2, then we have $q_\iota \xrightarrow{[w_1 \cdot a/w' \cdot w'_2 \cdot a']} q$, which concludes.
- If $w'_1 = \epsilon$, then $a' \in Last_1(q_1)$. By induction hypothesis on w_1 there is some w'_2 such that $q_\iota \xrightarrow{[w_1/w'_2 \cdot a']} q_1$. By transitivity we obtain $q_\iota \xrightarrow{[w_1 \cdot a/w'_2 \cdot a']} q$, which concludes.

Now for the right-to-left inclusion of Point 1, take q and w' such that $q_\iota \xrightarrow{[w_1 \cdot a/w']} q$. Necessarily there exist w'_1, w'_2 and q_1 such that $q_\iota \xrightarrow{[w_1/w'_1]} q_1$, $q_1 \xrightarrow{[a/w'_2]} q$ and $w'_1 \cdot w'_2 = w'$. By induction hypothesis $q_1 \in S_1$, so by definition of δ, $q \in S$. For the right-to-left inclusion of Point 2, take $q \in S$, and take a' and w' such that $q_\iota \xrightarrow{[w_1 \cdot a/w' \cdot a']} q$. Again, necessarily there exist w'_1, w'_2 and q_1 such that $q_\iota \xrightarrow{[w_1/w'_1]} q_1$, $q_1 \xrightarrow{[a/w'_2]} q$ and $w'_1 \cdot w'_2 = w' \cdot a'$. By induction hypothesis $q_1 \in S_1$. We distinguish two cases.

- If $w'_2 = \epsilon$, then $w'_1 = w' \cdot a'$, hence $q_\iota \xrightarrow{[w_1/w' \cdot a']} q_1$. By induction hypothesis, $a' \in Last_1(q_1)$, so by definition of δ, because $q_1 \in S_1$ and $q_1 \xrightarrow{[a/\epsilon]} q$, we obtain $a' \in Last(q)$.
- If $w'_2 = w'_3 \cdot a'$ for some w'_3, then by definition of δ, because $q_1 \in S_1$, we have $a' \in Last(q)$.

□

We now prove that the information set automaton computes information sets:

Proposition 13. *For every word $w \in \Sigma^*$, $\delta(\alpha_\iota, w).I = I_{\rightsquigarrow}(w)$.*

Proof. Let $w \in \Sigma^*$, and let $(S, Last) = \delta(\alpha_\iota, w)$. We remind that $I_{\rightsquigarrow}(w) = \{a \in \Sigma \mid \exists w' \cdot a \in \Sigma^*,\ w \rightsquigarrow w' \cdot a\}$ (Definition 29).

We start with the left-to-right inclusion. Let $a \in (S, Last).I$. By definition, $a \in Last(q)$ for some $q \in S \cap F$. By Lemma 6, there exists $w' \in \Sigma^*$ such that $q_\iota \xrightarrow{[w/w' \cdot a]} q$, and because $q \in F$, we have that $(w, w' \cdot a) \in [T] = \rightsquigarrow$, hence $a \in I_{\rightsquigarrow}(w)$.

For the right-to-left inclusion, take $a \in I_{\rightsquigarrow}(w)$. There exists w' such that $w \rightsquigarrow w' \cdot a$. Since $\rightsquigarrow = [T]$, there exists $q \in F$ such that $q_\iota \xrightarrow{[w/w' \cdot a]} q$. By Lemma 6, $q \in S$, and $a \in Last(q)$. Since $q \in S \cap F$, $a \in (S, Last).I$. □

Information set bisimulation and K45NM relations.

We define a bisimilarity relation over the states of $\mathcal{A}^T$, the information set automaton for T. We call this relation the *information set bisimilarity*. Informally, two states of the automaton are information set bisimilar if, for every possible word, reading it starting from one state or the other leads to states containing the same information set. We use this notion to establish a property of information set automata for K45NM relations that is central in our elementary procedure for the fully-uniform strategy with such relations (see Theorem 10).

Definition 31 (Information set bisimilarity)**.** Two states $\alpha, \alpha' \in A$ are *information set bisimilar*, written $\alpha \leftrightarrows_I \alpha'$, if for all $w \in \Sigma^*, \delta(\alpha, w).I = \delta(\alpha', w).I$. We call $\leftrightarrows_I$ the *information set bisimilarity* relation. Note that $\leftrightarrows_I$ is an equivalence relation. For a state α, $[\alpha]_{\leftrightarrows_I}$ denotes the equivalence class of α.

The following lemma is the crucial point that makes the fully-uniform strategy problem elementary for K45NM relations (Theorem 10). Informally, it states that if a relation is K45NM, then two related words bring the information set automaton to information set bisimilar states.

Lemma 7. *If $\rightsquigarrow$ is a* K45NM *relation then for all $w, w' \in \Sigma^*$, $w \rightsquigarrow w'$ implies that $\delta(\alpha_\iota, w) \leftrightarrows_I \delta(\alpha_\iota, w')$.*

Proof. Let $\alpha = \delta(\alpha_\iota, w)$ and $\alpha' = \delta(\alpha_\iota, w')$, and take $w'' \in \Sigma^*$. We prove that α and α' are information set bisimilar, *i.e.* for all $w'' \in \Sigma^*$, $\delta(\alpha, w'').I = \delta(\alpha', w'').I$. Take some $w'' \in \Sigma^*$. First, because $w \rightsquigarrow w'$ and $\rightsquigarrow$ satisfies No Miracles, $w \cdot w'' \rightsquigarrow w' \cdot w''$.

$\subseteq$: Let $a \in \delta(\alpha, w'').I = \delta(\alpha_\iota, w \cdot w'').I$. By Proposition 13, $a \in I(w \cdot w'')$, so there exists $u \in \Sigma^*$ such that $w \cdot w'' \rightsquigarrow u \cdot a$. We also have that $w \cdot w'' \rightsquigarrow w' \cdot w''$, and $\rightsquigarrow$ is Euclidean, so $w' \cdot w'' \rightsquigarrow u \cdot a$. It follows that $a \in I(w' \cdot w'')$, and by Proposition 13 again, $a \in \delta(\alpha_\iota, w' \cdot w'').I = \delta(\alpha', w'').I$.

$\supseteq$: Let $a \in \delta(\alpha', w'').I = \delta(\alpha_\iota, w' \cdot w'').I$. By Proposition 13, $a \in I(w' \cdot w'')$, so there exists $u \in \Sigma^*$ such that $w' \cdot w'' \rightsquigarrow u \cdot a$. We have that $w \cdot w'' \rightsquigarrow w' \cdot w''$, and $\rightsquigarrow$ is transitive, so $w \cdot w'' \rightsquigarrow u \cdot a$, and therefore $a \in I(w \cdot w'')$. By Proposition 13, $a \in \delta(\alpha_\iota, w \cdot w'').I = \delta(\alpha, w'').I$. □

We now show that quotienting the information set automaton $\mathcal{A}^T$ with $\leftrightarrows_I$ yields an automaton that still computes the information sets for T.

Definition 32. The *quotient automaton* is $\mathcal{A}^T_{\leftrightarrows_I} = (\Sigma, A_{\leftrightarrows_I}, \delta_{\leftrightarrows_I}, [\alpha_\iota]_{\leftrightarrows_I})$, where:

- $A_{\leftrightarrows_I}$ is the set of equivalence classes of $\leftrightarrows_I$,
- for $a \in \Sigma$, $\delta_{\leftrightarrows_I}([\alpha]_{\leftrightarrows_I}, a) = [\delta(\alpha, a)]_{\leftrightarrows_I}$, and
- $[\alpha]_{\leftrightarrows_I}.I = \alpha.I$.

Lemma 8. *$\mathcal{A}^T_{\leftrightarrows_I}$ is well defined and can be computed in time $O(|\mathcal{A}^T|^2)$.*

Proof. For $\alpha, \alpha' \in A$ such that $\alpha \leftrightarrows_I \alpha'$, we have that $\alpha.I = \alpha'.I$, so the information set of a state $[\alpha]_{\leftrightarrows_I}$ is well defined. Also for $\alpha, \alpha' \in A$ and $a \in \Sigma$, $\delta(\alpha, a) \leftrightarrows_I \delta(\alpha', a)$, hence $[\delta(\alpha, a)]_{\leftrightarrows_I} = [\delta(\alpha', a)]_{\leftrightarrows_I}$ and $\delta_{\leftrightarrows_I}$ is well defined.

The relation $\leftrightarrows_I$ can be computed in time $O(|\mathcal{A}^T|^2)$ (Kanellakis and Smolka, 1990), and from $\leftrightarrows_I$ the quotient automaton $\mathcal{A}^T_{\leftrightarrows_I}$ is computed in linear time. □

The following lemma is easily proved by induction:

Lemma 9. *For all $w \in \Sigma^*$, $\delta_{\leftrightarrows_I}([\alpha_\iota]_{\leftrightarrows_I}, w) = [\delta(\alpha_\iota, w)]_{\leftrightarrows_I}$.*

It follows that the quotient automaton still computes information sets correctly:

Proposition 14. *For all $w \in \Sigma^*$, $\delta_{\leftrightarrows_I}([\alpha_\iota]_{\leftrightarrows_I}, w).I = \delta(\alpha_\iota, w).I$.*

Restrained information sets.

In Section 5.3, we use information set automata to solve the fully-uniform strategy problem. They enable us to compute the set of positions where a formula φ should be evaluated in order to give a formula of the form $\boxminus\varphi$ a truth value. According to the semantics of the full quantifier, the information set that we aim at computing given a finite path is the set of last positions of related *paths*. But in general, given an instance $(\mathcal{G}, T, \Phi)$ of the problem, $[T]$ may relate some paths in $\mathcal{G}$ to words that do not correspond to any path. This is due to the fact that, as discussed after Definition 27 (page 64), the relation, and therefore the transducer T, may be defined independently from the arena $\mathcal{G}$.

We first describe how a simple modification of T enables us to compute the correct information sets, taking into account only those related words that are actual paths in $\mathcal{G}$. First, observe that $Paths_*$ is a regular language, and one can easily derive from $\mathcal{G}$ a finite word automaton that accepts it.

For the rest of this section, we fix a transducer T over some alphabet Σ and a word automaton $\mathcal{A}$ over Σ.

Definition 33. The *restriction* of T to $\mathcal{A}$ is the transducer $T|\mathcal{A} = T \circ T_{\mathcal{A}}$, where $T_{\mathcal{A}}$ is the identity transducer over $\mathcal{L}(\mathcal{A})$ (see Definition 14, page 27).

From now on we let $\rightsquigarrow = [T]$ and $\rightsquigarrow_{\mathcal{A}} = [T|\mathcal{A}]$.

Lemma 10. *It holds that:*

1. $\rightsquigarrow_{\mathcal{A}} = \rightsquigarrow \cap (\Sigma^* \times \mathcal{L}(\mathcal{A}))$
2. *for all* $w \in \Sigma^*$, $I_{\rightsquigarrow_{\mathcal{A}}}(w) = \{a \in \Sigma \mid \exists w' \cdot a \in \mathcal{L}(\mathcal{A})$ *such that* $w \rightsquigarrow w' \cdot a\}$.

Proof. Point 1 follows from the definition of $T|\mathcal{A}$, and Point 2 is a consequence of Point 1 and the definition of the information set (Definition 29). □

We now establish a variant of Lemma 7. It is clear that transitivity and Euclideanity are preserved by restriction, *i.e.* if $\rightsquigarrow$ is transitive and Euclidean, so is $\rightsquigarrow_{\mathcal{A}}$. It is not the case however of the No Miracles property. Indeed, assume that $\rightsquigarrow$ verifies No Miracles, and take $w, w' \in \Sigma^*$ such that $w \rightsquigarrow_{\mathcal{A}} w'$, and let $w'' \in \Sigma^*$. By definition, $\rightsquigarrow_{\mathcal{A}} \subseteq \rightsquigarrow$, hence $w \rightsquigarrow w'$, and because $\rightsquigarrow$ verifies No Miracles, we have that $w \cdot w'' \rightsquigarrow w' \cdot w''$. But if $w' \cdot w'' \notin \mathcal{L}(\mathcal{A})$, $w \cdot w'' \not\rightsquigarrow_{\mathcal{A}} w' \cdot w''$.

So even if $\rightsquigarrow$ is in K45NM, it may not be the case of $\rightsquigarrow_{\mathcal{A}}$. However, we prove that for $\mathcal{A}^{T|\mathcal{A}}$ to reach information set bisimilar states when reading two related words, it is not necessary for $[T|\mathcal{A}] = \rightsquigarrow_{\mathcal{A}}$ to be in K45NM, but it is sufficient if $[T] = \rightsquigarrow$ is. Let $\mathcal{A}^{T|\mathcal{A}} = (\Sigma, A, \delta, \alpha_\iota)$ be the information set automaton for $T|\mathcal{A}$.

Lemma 11. *If* $\rightsquigarrow$ *is a* K45NM *relation, then for all* $w, w' \in \Sigma^*$, $w \rightsquigarrow w'$ *implies that* $\delta(w) \leftrightarrows_I \delta(w')$.

Proof. The proof is almost the same as for Lemma 7, we just have in addition to verify whether a related word is in $\mathcal{L}(\mathcal{A})$ or not.

Let $\alpha = \delta(\alpha_\iota, w)$ and $\alpha' = \delta(\alpha_\iota, w')$, and take $w'' \in \Sigma^*$. We prove that α and α' are information set bisimilar, *i.e.* for all $w'' \in \Sigma^*$, $\delta(\alpha, w'').I = \delta(\alpha', w'').I$. Take some $w'' \in \Sigma^*$. First, because $w \rightsquigarrow w'$ and $\rightsquigarrow$ satisfies No Miracles, we have that $w \cdot w'' \rightsquigarrow w' \cdot w''$.

$\subseteq$: Let $a \in \delta(\alpha, w'').I = \delta(\alpha_\iota, w \cdot w'').I$. By Proposition 13, $a \in I_{\rightsquigarrow_\mathcal{A}}(w \cdot w'')$, so by Lemma 10, there exists $u \in \Sigma^*$ such that $u \cdot a \in \mathcal{L}(\mathcal{A})$ and $w \cdot w'' \rightsquigarrow u \cdot a$. We also have that $w \cdot w'' \rightsquigarrow w' \cdot w''$, and $\rightsquigarrow$ is Euclidean, so $w' \cdot w'' \rightsquigarrow u \cdot a$. Because $u \cdot a \in \mathcal{L}(\mathcal{A})$, we also have that $w' \cdot w'' \rightsquigarrow_\mathcal{A} u \cdot a$, and therefore $a \in I_{\rightsquigarrow_\mathcal{A}}(w' \cdot w'')$. By Proposition 13 again, $a \in \delta(\alpha_\iota, w' \cdot w'').I = \delta(\alpha', w'').I$.

$\supseteq$: Let $a \in \delta(\alpha', w'').I = \delta(\alpha_\iota, w' \cdot w'').I$. By Proposition 13, $a \in I_{\rightsquigarrow_\mathcal{A}}(w' \cdot w'')$, so there exists $u \in \Sigma^*$ such that $u \cdot a \in \mathcal{L}(\mathcal{A})$ and $w' \cdot w'' \rightsquigarrow u \cdot a$. We have that $w \cdot w'' \rightsquigarrow w' \cdot w''$, and $\rightsquigarrow$ is transitive, so $w \cdot w'' \rightsquigarrow u \cdot a$, and because $u \cdot a \in \mathcal{L}(\mathcal{A})$, we also have that $w \cdot w'' \rightsquigarrow_\mathcal{A} u \cdot a$, and therefore $a \in I_{\rightsquigarrow_\mathcal{A}}(w \cdot w'')$. By Proposition 13 again, $a \in \delta(\alpha_\iota, w \cdot w'').I = \delta(\alpha, w'').I$. □

We now turn to the proofs of Theorems 9 and 10.

5.3 Upper bounds

We establish the upper bounds for Theorem 9 and Theorem 10. We first observe that in the degenerate case FUS_0, the formula $\Phi \in \mathcal{FL}^0_{\rightsquigarrow}$ being a CTL^* formula, the transducer is irrelevant, and in fact the problem FUS_0 is exactly the problem of solving games with CTL^* winning condition. This problem is known to be 2-Exptime-complete (Kupferman and Vardi, 1997), and we rephrase the precise upper-bound in our context.

Proposition 15. *Let $\mathcal{G}$ be a 2^{AP}-labelled arena and $\Phi \in \mathsf{CTL}^*$. Letting d be the maximum branching degree in $\mathcal{G}$, solving the CTL^* game $(\mathcal{G}, \Phi)$ can be done in time $(|\mathcal{G}|^d \cdot 2^{|AP|})^{2^{O(|\Phi|)}}$.*

Proof. The proof is a reformulation of Kupferman and Vardi (1997) and is very similar to the one of Proposition 12. The only difference is that the formula being here a CTL^*-formula and not an $\mathcal{L}_{\rightsquigarrow}$-formula, we build an alternating automaton that accepts the models of Φ instead of a jumping automaton. □

Remark 10. With the same argument as in Remark 8, the complexity is the same when generalized strategies are considered.

5.3.1 The general case

Let $k > 0$ be a positive natural number, and let us fix for this section an instance $(\mathcal{G}, T, \Phi)$ of FUS_k over a set of atomic propositions AP, where $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$, and let us note once more $\Sigma = 2^{AP}$ and $\rightsquigarrow = [T]$. We describe a powerset construction that, relying strongly on the information set automaton, builds an instance of FUS_{k-1} of exponential size that is equivalent to $(\mathcal{G}, T, \Phi)$ regarding the existence of uniform strategies. Iterating this powerset construction yields an equivalent instance of FUS_0, which can be solved in time doubly exponential in the size of the formula (Proposition 15). In addition, a winning strategy in the latter CTL^* game straightforwardly provides a uniform strategy in the original instance.

Because the semantics of the full quantifier depends only on the universe and the binary relation, and not on the particular strategy Φ is evaluated on, we can use a bottom-up evaluation process in the formula before addressing the existence of a strategy. Informally, like in the classic powerset construction for games with imperfect information (Reif, 1984),

we build an arena with information sets in the positions, in which formulas of the form $\boxminus\varphi$ can be evaluated positionally if $\varphi \in \mathsf{CTL}^*$. According to the semantics of $\boxminus$, after a finite play, the information set that we require is the set of last positions of related *finite paths in the universe*.

First, observe that T works on alphabet Σ, hence using $\mathcal{A}^T$ we would get sets of valuations instead of sets of positions. We remedy this technical problem by building a transducer T_V that recognizes the relation on V^* induced by $\rightsquigarrow$. Formally, $T_V = T_{V\to\Sigma} \circ T \circ T_{\Sigma\to V}$, where $T_{V\to\Sigma}$ is a one-state deterministic transducer with $|V|$ transitions that outputs the valuations of the positions it reads and $T_{\Sigma\to V}$ is its (nondeterministic) inverse of same size. We obtain $[T_V] = [T_{V\to\Sigma}] \circ [T] \circ [T_{\Sigma\to V}]$, *i.e.* $[T_V] = \{(w, w') \mid w, w' \in V^*$ and $\mu(w) \rightsquigarrow \mu(w')\}$, and $|T_V| = |T_{V\to\Sigma}| \cdot |T| \cdot |T_{\Sigma\to V}| = O(|T| \cdot |\mathcal{G}|^2)$. From now on, for $w, w' \in V^*$, we may write $w \rightsquigarrow w'$ for $\mu(w) \rightsquigarrow \mu(w')$, *i.e.* $w[T_V]w'$.

The second technical problem is, as mentioned before Definition 33, page 70, that $[T_V]$ may relate arbitrary sequences of positions, while the information sets we want to compute only concern related paths of the universe $\mathcal{U} = Paths_*(V_\iota)$. To fix this, we take the restriction of T_V to $\mathcal{U}$, defining $T_\mathcal{G} = T_V|\mathcal{A}_\mathcal{G}$ (see Definition 33), where $\mathcal{A}_\mathcal{G}$ is a word automaton that recognizes the regular language $\mathcal{U}$ (easily obtained from $\mathcal{G}$). We have that $|T_\mathcal{G}| = |T_V| \cdot |\mathcal{A}_\mathcal{G}| = O(|T_V| \cdot |\mathcal{G}|) = O(|T| \cdot |\mathcal{G}|^3)$. Let $\mathcal{A}^{T_\mathcal{G}} = (V, A, \delta, \alpha_\iota)$ be the information set automaton for $T_\mathcal{G}$.

Lemma 12. *It holds that:*

1. $[T_\mathcal{G}] = [T_V] \cap (V^* \times \mathcal{U})$.
2. *for all* $w \in V^*$, $\delta(\alpha_\iota, w).I = \{v \mid \exists \rho \cdot v \in \mathcal{U}, w \rightsquigarrow \rho \cdot v\}$.

Proof. This is a direct application of Lemma 10 to $T_\mathcal{G}$. □

We describe the synchronization of the arena $\mathcal{G}$ with $\mathcal{A}^{T_\mathcal{G}}$, which yields an arena $\widehat{\mathcal{G}}$ that has the same dynamics as $\mathcal{G}$, but in addition computes the information sets required to evaluate fully-quantified formulas.

Definition 34. Let $\widehat{\mathcal{G}} := (\widehat{V}, \widehat{E}, \widehat{V}_\iota, \widehat{v}_\iota, \widehat{\mu})$, with

- $\widehat{V} = V \times A$,
- if $(v, \alpha) \in \widehat{V}$ and $v \to v'$, then $(v, \alpha) \mathrel{\widehat{\to}} (v', \delta(\alpha, v'))$
- $\widehat{V}_\iota = \{(v, \delta(\alpha_\iota, v)) \mid v \in V_\iota\}$
- $\widehat{v}_\iota = (v_\iota, \delta(\alpha_\iota, v_\iota))$ and
- $\widehat{\mu}(v, \alpha) = \mu(v)$

Each path in $\mathcal{G}$ defines a unique path in $\widehat{\mathcal{G}}$, and vice versa. To avoid confusion we shall use a "hat" version of each notation when it refers to the powerset arena $\widehat{\mathcal{G}}$. Consider the following function:

$$f : Paths_\omega(V_\iota) \to \widehat{Paths}_\omega(\widehat{V}_\iota)$$
$$\pi \mapsto \widehat{\pi} \text{ where for each } i \geq 0, \widehat{\pi}[i] = (\pi[i], \delta(\alpha_\iota, \pi[0, i])).$$

Clearly, f is a bijection between $Paths_\omega(V_\iota)$ and $\widehat{Paths}_\omega(\widehat{V}_\iota)$, *i.e.* between universes $\mathcal{U}$ and $\widehat{\mathcal{U}}$. When π is given, we shall write $\widehat{\pi}$ for $f(\pi)$, and when $\widehat{\pi}$ is given, π shall denote $f^{-1}(\widehat{\pi})$, and similarly for finite paths.

The next step is to eliminate all subformulas of Φ of the form $\boxminus\varphi$ with $d(\varphi) = 0$, *i.e.* $\varphi \in \mathsf{CTL}^*$. For each such subformula $\boxminus\varphi$ and each position $\widehat{v} = (v, \alpha)$ of $\widehat{\mathcal{G}}$, we model-check φ in all positions of the information set $\alpha.I$. Since φ holds in all these positions iff $\boxminus\varphi$ holds in (every finite path ending in) $\widehat{v}$, we can mark $\widehat{v}$ with a fresh atomic proposition $p_{\boxminus\varphi}$ when appropriate. This procedure, which we shall refer to as the *marking phase*, is described in Algorithm 1. From now on, $\widehat{\mathcal{G}}$ refers to the powerset arena after its labelling has been enriched by this marking phase.

```
1 foreach ⊟φ ∈ Sub(Φ) such that d(φ) = 0 do
2 |   foreach v̂ = (v, α) ∈ V̂ do
3 |   |   if ∀v' ∈ α.I, Paths_*(v') ⊨ φ then
4 |   |   |   μ̂(v̂) := μ̂(v̂) ∪ {p_{⊟φ}};
5 |   |   end
6 |   end
7 end
```

Algorithm 1: Marking the positions of $\widehat{\mathcal{G}}$.

For a state formula φ, we define $\widehat{\varphi}$ as the formula obtained by replacing each innermost subformula of the form $\boxminus\varphi'$ with $p_{\boxminus\varphi'}$, and similarly for path formulas. For example, if $\varphi = \boxminus p \wedge \mathbf{AG}\boxminus\mathbf{EX}\boxminus q$, then $\widehat{\varphi} = p_{\boxminus p} \wedge \mathbf{AG}\boxminus\mathbf{EX}p_{\boxminus q}$.

It just remains to define the transducer $\widehat{T}$ such that $\rho \rightsquigarrow \rho'$ if and only if $\widehat{\rho} \mathrel{\widehat{\rightsquigarrow}} \widehat{\rho}'$, where $\mathrel{\widehat{\rightsquigarrow}} = [\widehat{T}]$. By Definition 34, a position (v, α) in $\widehat{\mathcal{G}}$ has the same valuation as the underlying position v in $\mathcal{G}$, except for the fresh atomic propositions added during the marking phase. So, letting $\widehat{AP}$ be the set AP augmented with these fresh atomic propositions, we just modify T so that it works on alphabet $2^{\widehat{AP}}$ but ignores these additional propositions. Thus we define $\widehat{T} = T_{2^{\widehat{AP}} \to 2^{AP}} \circ T \circ T_{2^{AP} \to 2^{\widehat{AP}}}$, where $T_{2^{\widehat{AP}} \to 2^{AP}}$ is a one state deterministic transducer that reads valuations on $\widehat{AP}$ and outputs underlying valuations on AP by erasing fresh propositions, and $T_{2^{AP} \to 2^{\widehat{AP}}}$ is its inverse. The number of fresh atomic propositions is bounded by $|\Phi|$, so $|\widehat{T}| = O(2^{|\widehat{AP}|} \cdot |T| \cdot 2^{|\widehat{AP}|}) = |T| \cdot 2^{O(|AP|)} \cdot 2^{O(|\Phi|)}$, and letting $\mathrel{\widehat{\rightsquigarrow}} = [\widehat{T}]$ we have that for $\rho, \rho' \in Paths_*(V_\iota)$, $\rho \rightsquigarrow \rho'$ if and only if $\widehat{\rho} \mathrel{\widehat{\rightsquigarrow}} \widehat{\rho}'$.

Since $d(\widehat{\Phi}) = d(\Phi) - 1 = k - 1$, $(\widehat{\mathcal{G}}, \widehat{T}, \widehat{\Phi})$ is an instance of FUS_{k-1}, and we prove that:

Proposition 16. *$(\mathcal{G}, T, \Phi) \in \mathsf{FUS}_k$ if, and only if, $(\widehat{\mathcal{G}}, \widehat{T}, \widehat{\Phi}) \in \mathsf{FUS}_{k-1}$.*

We establish Lemma 13 below, and Proposition 16 follows from the fact that the bijection f between $Paths_\omega(V_\iota)$ and $\widehat{Paths_\omega}(\widehat{V}_\iota)$ induces a bijection between strategy trees in $\mathcal{G}$ and strategy trees in $\widehat{\mathcal{G}}$.

More precisely, f induces a bijection between the trees of paths in $\mathcal{G}$ and those in $\widehat{\mathcal{G}}$. For any $t \subseteq Paths_*(v)$ where $v \in V_\iota$, we define $\widehat{t} \subseteq \widehat{Paths_*}(v, \delta(\alpha_\iota, v))$ by $\widehat{t} = \{\widehat{\rho} \mid \rho \in t\}$, with labelling $\widehat{\ell}(\widehat{\rho}) = \widehat{\mu}(\mathrm{last}(\widehat{\rho}))$. Notice that a node $x \in t$ being a finite path, $\widehat{x}$ is defined.

Lemma 13. *For any state formula $\varphi \in Sub(\Phi)$ and for any labelled tree $t \subseteq Paths_*(V_\iota)$, we have: for all $x \in t$, $t, x \models \varphi$ if, and only if, $\widehat{t}, \widehat{x} \models \widehat{\varphi}$.*

Proof. The proof is done by induction on φ. For a branch $\lambda = x_0 x_1 \ldots$, we let $\widehat{\lambda} = \widehat{x}_0 \widehat{x}_1 \ldots$

Case $\varphi = p$. Let $t \subseteq Paths_*(v)$ for some $v \in V_\iota$, and let $x \in t$. Letting $x = v_0 \dots v_n$, by definition of f we have $\widehat{x} = \widehat{v}_0 \dots \widehat{v}_n$, where for $0 \leq i \leq n$, $\widehat{v}_i = (v_i, \delta(\alpha_\iota, v_0 \dots v_i))$. By Definition 34 of $\widehat{\mathcal{G}}$, $\widehat{v}_n$ is labelled with the same atomic propositions as v_n, except for the fresh ones added in the marking phase (Algorithm 1). But because p is a subformula of Φ, it cannot be fresh, thus p is in the label of x if and only if p is in the label of $\widehat{x}$, and the result follows.

Cases $\varphi = \varphi_1 \vee \varphi_2$ and $\varphi = \neg\varphi'$. Trivial by induction hypothesis, noting that $\widehat{\varphi_1 \vee \varphi_2} = \widehat{\varphi_1} \vee \widehat{\varphi_2}$, and $\widehat{\neg\varphi'} = \neg\widehat{\varphi'}$.

Case $\varphi = \mathbf{A}\psi_0$. We prove the following lemma:

> **Lemma 14.** *For any path formula $\psi \in Sub(\varphi)$, for any labelled tree $t \subseteq Paths_*(V_\iota)$, we have: for all $\lambda \in Branches(t)$, $t, \lambda \models \psi$ if and only if $\widehat{t}, \widehat{\lambda} \models \widehat{\psi}$.*
>
> *Proof.* The proof is by induction on ψ.
>
> **Case $\psi = \varphi'$.** Let $t \subseteq Paths_*(v)$ for some $v \in V_\iota$, and let $\lambda \in Branches(t)$. Let $x = \lambda[0]$. By definition, $t, \lambda \models \varphi'$ iff $t, x \models \varphi'$, and $\widehat{t}, \widehat{\lambda} \models \widehat{\varphi'}$ iff $\widehat{t}, \widehat{x} \models \widehat{\varphi'}$. Furthermore, because φ' is a subformula of $\varphi = \mathbf{A}\psi_0$, we have by induction hypothesis that $t, x \models \varphi'$ if and only if $\widehat{t}, \widehat{x} \models \widehat{\varphi'}$, which concludes.
>
> **Cases $\psi = \neg\psi'$, $\psi_1 \vee \psi_2$, $\mathbf{X}\psi'$, $\psi_1 \mathbf{U} \psi_2$.** Trivial by induction hypothesis.
>
> □

We can now prove the induction case $\varphi = \mathbf{A}\psi_0$. Let $t \subseteq Paths_*(v)$ for some $v \in V_\iota$, and let $x \in t$. By Lemma 14, for all $\lambda \in Branches(t)$, $t, \lambda \models \psi_0$ if and only if $\widehat{t}, \widehat{\lambda} \models \widehat{\psi_0}$. Because f induces a bijection between $Branches(x)$ and $\widehat{Branches}(\widehat{x})$, we obtain that $t, x \models \mathbf{A}\psi_0$ if and only if $\widehat{t}, \widehat{x} \models \mathbf{A}\widehat{\psi_0}$. We conclude by noticing that $\mathbf{A}\widehat{\psi_0} = \widehat{\mathbf{A}\psi_0}$.

Case $\varphi = \boxdot\varphi'$. Let $t \subseteq Paths_*(v)$ for some $v \in V_\iota$, and let $x \in t$. Letting $x = v_0 \dots v_n$, we have by definition of f that $\widehat{x} = \widehat{v}_0 \dots \widehat{v}_n$, where for $0 \leq i \leq n$, $\widehat{v}_i = (v_i, \delta(\alpha_\iota, v_0 \dots v_i))$. So $\widehat{v}_n = (v_n, \alpha_n)$ where $\alpha_n = \delta(\alpha_\iota, x)$.

If $d(\varphi') = 0$, then $\widehat{\varphi} = p_\varphi$. The definition of $\models$ gives this first equivalence:

$$\widehat{t}, \widehat{x} \models p_\varphi \iff p_\varphi \in \widehat{\mu}(\widehat{v}_n). \tag{5.1}$$

Considering the marking phase (Algorithm 1), we have:

$$p_\varphi \in \widehat{\mu}(\widehat{v}_n) \iff \forall v' \in \alpha_n.I,\ Paths_*(v') \models \varphi'. \tag{5.2}$$

By Lemma 12, and because $\alpha_n = \delta(\alpha_\iota, x)$, $v' \in \alpha_n.I$ if and only if there is a node $y \in \mathcal{U}$ (recall that $\mathcal{U} = Paths_*(V_\iota)$) such that $x \rightsquigarrow y$ and $\mathrm{last}(y) = v'$. Combining this with Equations (5.1) and (5.2) yields:

$$\widehat{t}, \widehat{x} \models p_\varphi \iff \forall y \in \mathcal{U} \text{ s.t. } x \rightsquigarrow y,\ Paths_*(\mathrm{last}(y)) \models \varphi'. \tag{5.3}$$

Observe that for a node $y \in \mathcal{U}$, $Paths_*(\mathrm{last}(y))$ is the same tree as the subtree of $\mathcal{U}_y$ rooted in y; also, φ' contains no $\boxdot$ quantifier but is a pure CTL^* formula,

hence its evaluation in a node of a tree does not depend on the path leading to this node, but only on the subtree from this node. From this and Equation (5.3) we have:

$$\widehat{t}, \widehat{x} \models p_\varphi \iff \forall y \in \mathcal{U} \text{ s.t. } x \rightsquigarrow y,\ \mathcal{U}_y, y \models \varphi'.$$

The semantics of $\boxdot$ gives the requested equivalence:

$$\widehat{t}, \widehat{x} \models \widehat{\varphi} \iff t, x \models \varphi.$$

If $d(\varphi') > 0$**,** then $\widehat{\varphi} = \boxdot\widehat{\varphi'}$. Let $y \in \mathcal{U}$. First, because $\mathcal{U} = Paths_*(V_\iota)$, we have that $\mathcal{U}_y \subseteq Paths_*(v)$ for some $v \in V_\iota$, namely $v = y[0]$. So by induction hypothesis, $\mathcal{U}_y, y \models \varphi'$ if and only if $\widehat{\mathcal{U}_y}, \widehat{y} \models \widehat{\varphi'}$. Second, by definition of $\widehat{T}$, $x \rightsquigarrow y$ if and only if $\widehat{x} \mathrel{\widehat{\rightsquigarrow}} \widehat{y}$. Noticing that $\widehat{\mathcal{U}_y} = \widehat{\mathcal{U}}_{\widehat{y}}$, we obtain that $t, x \models \boxdot\varphi'$ if, and only if, $\widehat{t}, \widehat{x} \models \boxdot\widehat{\varphi'}$.

□

Lemma 13 shows that each powerset construction yields an equivalent FUS instance with a strictly smaller $\boxdot$-depth, and iterating the process we obtain an equivalent CTL* game that it remains to solve, which is a 2-Exptime-complete problem (Kupferman and Vardi, 1997) (see also Proposition 15).

We can now establish the upper bound for Theorem 9. For convenience, we introduce iterated exponential functions as follows:

Definition 35. For all $k, n \in \mathbb{N}$, $\exp^0(n) = n$ and $\exp^{k+1}(n) = 2^{\exp^k(n)}$.

Proposition 17. *Let $(\mathcal{G}, T, \Phi)$ be an instance of FUS_k for some $k \geq 0$, and note d the maximum branching degree in $\mathcal{G}$. Deciding whether $(\mathcal{G}, T, \Phi) \in \mathsf{FUS}_k$ can be done in time $\exp^k(|\mathcal{G}, T, \Phi|^{O(1)})^{d\cdot|AP|\cdot 2^{O(|\Phi|)}}$.*

Proof. The proof is by induction on k.

Case $k = 0$**.** Let $(\mathcal{G}, T, \Phi)$ be an instance of FUS_0. FUS_0 is exactly the strategy problem for Player 1 in CTL* games, and by Proposition 15 this problem can be solved in time $(|\mathcal{G}|^d \cdot 2^{|AP|})^{2^{O(|\Phi|)}}$, which is less than $\exp^0(|\mathcal{G}, T, \Phi|)^{d\cdot|AP|\cdot 2^{O(|\Phi|)}}$.

Case $k + 1$**.** Let $(\mathcal{G}, T, \Phi)$ be an instance of FUS_{k+1}, with $k \geq 0$. By Proposition 16, deciding whether $(\mathcal{G}, T, \Phi) \in \mathsf{FUS}_{k+1}$ is equivalent to deciding whether $(\widehat{\mathcal{G}}, \widehat{T}, \widehat{\Phi}) \in \mathsf{FUS}_k$, which by induction hypothesis can be done in time $\exp^k(|\widehat{\mathcal{G}}, \widehat{T}, \widehat{\Phi}|^{O(1)})^{d\cdot|\widehat{AP}|\cdot 2^{O(|\Phi|)}}$, where d is the maximum branching degree in $\widehat{\mathcal{G}}$. Observe that by construction of $\widehat{\mathcal{G}}$, d is also the maximum branching degree in $\mathcal{G}$. Observe also that the number of fresh atomic propositions used in the marking phase is bounded by $|\Phi|$, thus $|\widehat{AP}| \leq |AP| + |\Phi|$ and $\exp^k(|\widehat{\mathcal{G}}, \widehat{T}, \widehat{\Phi}|^{O(1)})^{d\cdot|\widehat{AP}|\cdot 2^{O(|\Phi|)}}$ is in $\exp^k(|\widehat{\mathcal{G}}, \widehat{T}, \widehat{\Phi}|^{O(1)})^{d\cdot|AP|\cdot 2^{O(|\Phi|)}}$. We prove that the instance $(\widehat{\mathcal{G}}, \widehat{T}, \widehat{\Phi})$ is of size $|\widehat{\mathcal{G}}, \widehat{T}, \widehat{\Phi}| = 2^{|\mathcal{G}, T, \Phi|^{O(1)}}$, hence solving it takes time $\exp^{k+1}(|\mathcal{G}, T, \Phi|^{O(1)})^{d\cdot|AP|\cdot 2^{O(|\Phi|)}}$. We also prove that computing the powerset instance $(\widehat{\mathcal{G}}, \widehat{T}, \widehat{\Phi})$ takes time less than $2^{|\mathcal{G}, T, \Phi|^{O(1)}}$, so that the decision procedure runs in time $\exp^{k+1}(|\mathcal{G}, T, \Phi|^{O(1)})^{d\cdot|AP|\cdot 2^{O(|\Phi|)}}$.

First, by Lemma 5, $\mathcal{A}^{T_\mathcal{G}}$ is of size $2^{O(|T_\mathcal{G}|\cdot|\Sigma|)}$, which is also the time needed to compute it. Because $T_\mathcal{G}$ works on alphabet $\Sigma = V$ and is of size $|T| \cdot |\mathcal{G}|^3$, we have that $|\mathcal{A}^{T_\mathcal{G}}| = 2^{O(|T_\mathcal{G}|\cdot|\Sigma|)} = 2^{(|\mathcal{G}|+|T|)^{O(1)}}$. Then, computing $\widehat{\mathcal{G}}$ from $\mathcal{G}$ and $\mathcal{A}^{T_\mathcal{G}}$ takes time $|\mathcal{G}| \cdot |\mathcal{A}^{T_\mathcal{G}}| = 2^{(|\mathcal{G}|+|T|)^{O(1)}}$, which is also the size of $\widehat{\mathcal{G}}$. Performing the marking phase requires to model-check at most $|\Phi|$ CTL^* formulas $|\mathcal{G}|$ times. Model-checking CTL^* is in PSPACE hence in EXPTIME, thus the marking phase can be done in time $2^{(|\mathcal{G}|+|\Phi|)^{O(1)}}$. Finally, we have seen that $|\widehat{T}| \leq |T| \cdot 2^{O(|AP|)} \cdot 2^{O(|\Phi|)}$, and it is also the time needed to compute it. Summing up everything yields the required results.

□

By Proposition 17, the complexity bounds are as follows:

- FUS_0 can be solved in time $|\mathcal{G}, T, \Phi|^{d\cdot|AP|\cdot 2^{O(|\Phi|)}}$,
- FUS_1 can be solved in time $2^{|\mathcal{G},T,\Phi|^{O(1)}\cdot 2^{O(|\Phi|)}}$ (because $d \leq |\mathcal{G}|$ and $|AP| \leq |\mathcal{G}, T, \Phi|$),
- and for $k \geq 2$, FUS_k can be solved in time $\exp^k(|\mathcal{G}, T, \Phi|^{O(1)})$.

This establishes the upper bounds for Theorem 9.

5.3.2 The elementary case

We prove Theorem 10 which considers rational binary relations in K45NM. For such relations, all the $\boxminus$ quantifiers can be eliminated in the formula with a single powerset construction.

Consider the marking phase described in Algorithm 1. To evaluate whether $\boxminus\varphi$ holds in a position $\widehat{v}$ reached after some partial play $\widehat{\rho}$, we have to evaluate formula φ in all paths ρ' such that $\rho \rightsquigarrow \rho'$. If φ does not contain any subformula of the form $\boxminus\varphi'$, *i.e.* φ is a CTL^* state formula, knowing the last position of each such ρ' is sufficient to evaluate φ; the information set contained in $\widehat{v}$ provides this information. But if φ has subformulas of the form $\boxminus\varphi'$, evaluating φ in a related path ρ' requires not only the last position of ρ', but also the state $\delta(\alpha_\iota, \rho')$ of $\mathcal{A}^{T_\mathcal{G}}$ after reading ρ'.

In the general case, as described in the proof of Proposition 17, we perform a new powerset construction, such that information sets are sets of positions of $\widehat{\mathcal{G}}$ (and no longer of $\mathcal{G}$), *i.e.* an element of an information set is a pair (v, α) where v is the last position of a related path, and α the state of $\mathcal{A}^{T_\mathcal{G}}$ after reading this path.

The reason why $\mathsf{FUS}_{\text{K45NM}}$ is elementary lies in the fact that, as established in Section 5.2, if $\rightsquigarrow$ is in K45NM and $\rho \rightsquigarrow \rho'$, then $\delta(\alpha_\iota, \rho)$ and $\delta(\alpha_\iota, \rho')$ are information set bisimilar. To take advantage of this we do not build $\widehat{\mathcal{G}}$ from $\mathcal{A}^{T_\mathcal{G}}$ directly, but we first quotient it according to Definition 32. Thus in Algorithm 1, when evaluating $\boxminus\varphi$ in a position $\widehat{v} = (v, \alpha)$ reached after some play $\widehat{\rho}$, we know that for each path ρ' such that $\rho \rightsquigarrow \rho'$, $\delta(\alpha_\iota, \rho') = \delta(\alpha_\iota, \rho) = \alpha$, and therefore we know that φ should be evaluated in position $\widehat{v}' = (\text{last}(\rho'), \alpha)$ of $\widehat{\mathcal{G}}$.

Let $(\mathcal{G}, T, \Phi)$ be an instance of $\mathsf{FUS}_{\text{K45NM}}$, note V the set of positions in $\mathcal{G}$ and $\rightsquigarrow = [T]$ (recall $\rightsquigarrow$ is a K45NM relation). We describe how, with one powerset construction only, we obtain an equivalent CTL^* game.

First, like in the general case, let us define T_V the transducer that relates two words in V^* if the corresponding sequences of valuations are related by T, and define $T_\mathcal{G}$ the restriction of T_V that only outputs words in $\mathcal{U} = Paths_*(V_\iota)$: $T_\mathcal{G} = T_V|\mathcal{A}_\mathcal{G}$, where $\mathcal{L}(\mathcal{A}_\mathcal{G}) = \mathcal{U}$.

Now we build the information set automaton $\mathcal{A}^{T_\mathcal{G}}$ for $T_\mathcal{G}$ and we quotient it by its bisimilarity relation $\leftrightarrows_I$. Let $\mathcal{A}^{T_\mathcal{G}}_{\leftrightarrows_I} = (V, A, \delta, \alpha_\iota)$ be this quotient information set automaton.

We first state that this quotient automaton computes the information we need. It follows directly from Lemma 12 page 72 and Proposition 14 page 69:

Lemma 15. *For all $w \in V^*$, $\delta(\alpha_\iota, w).I = \{v \mid \exists \rho \cdot v \in \mathcal{U}, w \rightsquigarrow \rho \cdot v\}$.*

Because $\rightsquigarrow$ is a K45NM relation and we consider the quotient automaton of $\mathcal{A}^{T_\mathcal{G}}$, Lemma 11 page 70 gives us:

Lemma 16. *For all $w, w' \in V^*$, $w \rightsquigarrow w'$ implies that $\delta(w) = \delta(w')$.*

We define the powerset arena as in Definition 34, except that we synchronize $\mathcal{G}$ with $\mathcal{A}^{T_\mathcal{G}}_{\leftrightarrows_I}$ instead of $\mathcal{A}^{T_\mathcal{G}}$. We note it $\widetilde{\mathcal{G}} = (\widetilde{V}, \widetilde{E}, \widetilde{V}_\iota, \widetilde{v}_\iota, \widetilde{\mu})$.

The bijection f between $\mathcal{U}$ and $\widetilde{\mathcal{U}}$ is defined as in the general case: for an infinite path π, $f(\pi) = \widetilde{\pi}$ where for each $i \geq 0$, $\widetilde{\pi}[i] = (\pi[i], \delta(\alpha_\iota, \pi[0, i]))$, and similarly for finite paths and for trees. To avoid confusion with the general case we denote the image of an object by f with a tilde instead of a hat ($\widetilde{\rho}$, $\widetilde{t}\ldots$).

For an $\mathcal{L}_\rightsquigarrow$ formula φ we define its *flattening* $\widetilde{\varphi}$ as the CTL^* formula obtained by recursively replacing each subformula $\boxminus\varphi'$ with $p_{\boxminus\varphi'}$, starting from innermost subformulas. For example,

$$\widetilde{\boxminus \mathbf{EX} \boxminus q} = p_{\boxminus \mathbf{EX} p_{\boxminus q}}.$$

We describe in Algorithm 2 the new marking procedure, which evaluates *all* subformulas of the form $\boxminus\varphi$, starting with the innermost ones, and at the same time computes the flattening of Φ. From now on, $\widetilde{\mathcal{G}}$ refers to the powerset arena after its labelling has been enriched by this marking phase.

```
1  Φ̃ := Φ;
2  while d(Φ̃) > 0 do
3  |  foreach ⊟φ ∈ Sub(Φ̃) such that d(φ) = 0 do
4  |  |  foreach ṽ = (v, α) ∈ Ṽ do
5  |  |  |  if ∀v' ∈ α.I, (v', α) ⊨ φ then
6  |  |  |  |  μ̃(ṽ) := μ̃(ṽ) ∪ {p_{⊟φ}};
7  |  |  |  end
8  |  |  end
9  |  |  Φ̃ := Φ̃[p_{⊟φ}/⊟φ]
10 |  end
11 end
```

Algorithm 2: Marking the positions of $\widetilde{\mathcal{G}}$.

Lemma 17. *For all state formula $\varphi \in Sub(\Phi)$, for all $v \in V_\iota$, for all labelled tree $t \subseteq Paths_*(v)$ and for all $x \in t$, $t, x \models \varphi$ if, and only if, $\widetilde{t}, \widetilde{x} \models \widetilde{\varphi}$.*

Proof. The proof is by induction on φ; we only treat the case $\varphi = \boxbslash\varphi'$, the others can be treated exactly as in the proof of Lemma 13. Let $t \subseteq Paths_*(v)$ for some $v \in V_\iota$, and take $x \in t$. Note $x = v_0 \dots v_n$ and $\widehat{x} = \widehat{v}_0 \dots \widehat{v}_n$. Letting $\alpha_n = \delta(\alpha_\iota, x)$, by definition of f we have that $\widehat{v}_n = (v_n, \alpha_n)$.

Because $\varphi = \boxbslash\varphi'$, $\widetilde{\varphi} = p_\varphi$. The definition of $\models$ gives this first equivalence:

$$\widetilde{t}, \widetilde{x} \models p_\varphi \iff p_\varphi \in \widetilde{\mu}(\widetilde{v}_n). \tag{5.4}$$

Consider the marking phase (Algorithm 2). When $\boxbslash\varphi'$ is evaluated on positions of $\widetilde{\mathcal{G}}$, fully-quantified subformulas of φ' (if any) have already been evaluated and replaced with fresh atomic propositions, so it is actually the flattening $\widetilde{\varphi'}$ of φ' that is evaluated in positions (v', α_n), where $v' \in \alpha_n.I$. This gives us:

$$p_\varphi \in \widetilde{\mu}(\widetilde{v}_n) \iff \forall v' \in \alpha_n.I,\ (v', \alpha_n) \models \widetilde{\varphi'}. \tag{5.5}$$

By Lemma 15, and because $\alpha_n = \delta(\alpha_\iota, x)$, $v' \in \alpha_n.I$ if and only if there is a node $y \in \mathcal{U}$ such that $x \rightsquigarrow y$ and $\mathrm{last}(y) = v'$. Combining this with equations 5.4 and 5.5 yields:

$$\widetilde{t}, \widetilde{x} \models p_\varphi \iff \forall y \in \mathcal{U} \text{ s.t. } x \rightsquigarrow y,\ (\mathrm{last}(y), \alpha_n) \models \widetilde{\varphi'}. \tag{5.6}$$

Now, by Lemma 16, we have that for all $y \in \mathcal{U}$ such that $x \rightsquigarrow y$, $\delta(\alpha_\iota, y) = \delta(\alpha_\iota, x) = \alpha_n$, and together with the definition of the bijection f this implies that $(\mathrm{last}(y), \alpha_n) = \mathrm{last}(\widetilde{y})$. With this, Equation 5.6 becomes:

$$\widetilde{t}, \widetilde{x} \models p_\varphi \iff \forall y \in \mathcal{U} \text{ s.t. } x \rightsquigarrow y,\ \mathrm{last}(\widetilde{y}) \models \widetilde{\varphi'}. \tag{5.7}$$

Because $\widetilde{\varphi'}$ contains no $\boxbslash$ quantifier but is a pure CTL^* formula, its evaluation in a node of a tree does not depend on the path leading to this node, but only on the subtree from this node. From this and Equation 5.7 we have that:

$$\widetilde{t}, \widetilde{x} \models p_\varphi \iff \forall y \in \mathcal{U} \text{ s.t. } x \rightsquigarrow y,\ \widetilde{\mathcal{U}_{\widetilde{y}}}, \widetilde{y} \models \widetilde{\varphi'}. \tag{5.8}$$

By induction hypothesis and because $\widetilde{\mathcal{U}_{\widetilde{y}}} = \widetilde{\mathcal{U}_y}$, we obtain:

$$\widetilde{t}, \widetilde{x} \models p_\varphi \iff \forall y \in \mathcal{U} \text{ s.t. } x \rightsquigarrow y,\ \mathcal{U}_y, y \models \varphi'.$$

We finally get:

$$\widetilde{t}, \widetilde{x} \models \widetilde{\boxbslash\varphi'} \iff t, x \models \boxbslash\varphi'.$$

□

Proposition 18. *Deciding* $(\mathcal{G}, T, \Phi) \in \mathsf{FUS}_{\mathrm{K45NM}}$ *can be done in time* $2^{|\mathcal{G},T,\Phi|^{O(1)}.2^{O(|\Phi|)}}$.

Proof. Let $(\mathcal{G}, T, \Phi)$ be an instance of $\mathsf{FUS}_{\mathrm{K45NM}}$. As detailed in the proof of Proposition 17, computing $\mathcal{A}^{T_\mathcal{G}}$ takes exponential time, and by Lemma 8 computing the quotient $\mathcal{A}^{T_\mathcal{G}}_{\leftrightarrows_I}$ takes quadratic time, so computing the powerset arena $\widetilde{\mathcal{G}}$ can be done in time $2^{|\mathcal{G},T,\Phi|^{O(1)}}$, and $|\widetilde{\mathcal{G}}| = 2^{|\mathcal{G},T,\Phi|^{O(1)}}$. The marking phase requires to perform the model checking of at most $|\Phi|$ CTL^* formulas of size $O(|\Phi|)$ on all the positions of $\widetilde{\mathcal{G}}$. The

model checking of a CTL^* formula of size m on an arena of size n can be done in time $2^{O(m)}O(n)$ (Kupferman et al., 2000), so each model checking here can be done in time $2^{O(|\Phi|)}O(|\widetilde{\mathcal{G}}, \widetilde{T}, \widetilde{\Phi}|)$, hence Algorithm 2 runs in time $2^{|\mathcal{G},T,\Phi|^{O(1)}}$. By Lemma 17, and because f induces a bijection between the strategy trees in $\mathcal{G}$ and those in $\widetilde{\mathcal{G}}$, solving $(\mathcal{G}, T, \Phi)$ is equivalent to solving the CTL^* game $(\widetilde{\mathcal{G}}, \widetilde{\Phi})$. By Proposition 15 this can be done in time $|\widetilde{\mathcal{G}}|^{d\cdot|AP|\cdot 2^{O(|\widetilde{\Phi}|)}}$, where d is the maximum branching degree of $\widetilde{\mathcal{G}}$. Because d is also the maximum branching degree of $\mathcal{G}$ it is bounded by $|\mathcal{G}|$, and by definition $|AP| \leq |\mathcal{G}, T, \Phi|$, hence solving the CTL^* game takes at most time $2^{|\mathcal{G},T,\Phi|^{O(1)}\cdot 2^{O(|\Phi|)}}$. Summing this to the time required to build the CTL^* game yields the result. □

Proposition 18 gives the upper bound for Theorem 10. Notice that the complexity is doubly exponential in the size of the formula, but for a fixed formula the problem falls in Exptime.

Synthesis of fully-uniform strategies. Regarding the problem of synthesizing uniform strategies, both in the general and the elementary case, the final step of the decision procedure consists in solving the final CTL^* game, and this is done by deciding the nonemptiness of a parity tree automaton that accepts the set of winning strategy trees. Like for the synthesis of strictly-uniform strategies with recognizable relations (see Page 59), the decision procedure can be adapted to synthesize a finite memory winning strategy when there is a winning strategy. Assuming that each position v in the original arena is identified with a dedicated atomic proposition p_v enables to keep track, in the successive powerset constructions, what original position underlies a powerset one. This way a regular winning strategy in the CTL^* game readily gives a fully-uniform strategy in the initial game. This finite memory strategy has the same number of states as the parity tree automaton. In the elementary case, this would be doubly exponential in the size of the formula. In the general case it would be the same for a $\boxminus$-depth below 2, and above it would be k-exponential in the size of the original problem.

5.4 Lower bounds

Regarding Theorem 10, we easily have:

Proposition 19. $\mathsf{FUS}_{\mathrm{K45NM}}$ *is* 2-Exptime*-hard.*

Proof. The problem of solving CTL^* games, which is 2-Exptime-complete (Kupferman et al., 2001), readily reduces to $\mathsf{FUS}_{\mathrm{K45NM}}$. □

We turn to the lower bounds for Theorem 9. First, FUS, like $\mathsf{FUS}_{\mathrm{K45NM}}$, contains the problem of solving CTL^* games, which gives the 2-Exptime lower bound for FUS_0 and FUS_1. It remains to prove the k-Exptime lower bounds for FUS_k with $k \geq 2$, which is done by the following proposition.

Proposition 20. *For $k \in \mathbb{N}$, FUS_{k+1} is $(k+1)$-*Exptime*-hard even if the $\mathcal{FL}^{k+1}_{\rightsquigarrow}$ formula is assumed to be fixed and the transducer is assumed to be synchronous.*

The proof of Proposition 20, by reduction of the word problem for exp[k]-space bounded *alternating* Turing Machines, is due to Laura Bozzelli. Because it is very technical, we do not include it here, but it can be found in Appendix A.

5.5 Conclusion and related work

In this chapter we proved that the existence of uniform strategies for uniformity properties that only use the full quantifier can be decided for the whole class of rational relations. The procedure is however nonelementary, and we proved that the height of the tower of exponentials matches precisely the number of nested quantifiers in the uniformity property. To establish the upper bounds we defined a notion of information set automaton that computes information sets for rational relations. The proof for the matching lower bounds, quite involved, is due to Laura Bozzelli. We also established that for rational relations that verify transitivity, Euclideanity and No Miracles, the uniform strategy problem is only 2-Exptime-complete. This result relies on a notion of information set bisimulation in information set automata.

It is important to remark that our framework is closely related to *rational graphs* as studied by Morvan (2000); Morvan and Rispal (2005). A graph is rational if its set of nodes is a regular word language, and each relation between nodes is a rational relation. Rational graphs thus form a class of infinite graphs that enjoy finite representations. Clearly, a finite arena together with a rational relation on finite plays is a rational graph with two relations. The nodes of the graph are the partial plays, which form a prefix-closed regular language accepted by the game arena. The first relation is the rational relation between plays that gives the semantics of $\boxdot$, and the second one, much simpler, is the successor relation on plays, that relates each partial play to all its extensions by one move. However, existing results on rational graphs could not help us to solve our uniform strategy problem. Indeed, the first order theory of rational graphs is known to be undecidable. For instance, properties like reflexivity, transitivity and symmetry, which are definable by very simple first order formulas, are undecidable for rational relations (Johnson, 1986). Morvan (2000) proves that even the formula $\exists x, x\mathbf{R}x$ is undecidable on rational graphs.

Restricting to synchronous transducers, *i.e.* regular relations, yields the class of models called *automatic graphs* that have been intensively studied (Khoussainov and Nerode, 1994; Blumensath and Grädel, 2000, 2004; Khoussainov et al., 2007). The model checking of first-order logic is decidable on automatic graphs (Blumensath and Grädel, 2004). However, the existence of a strategy in an infinite-duration game is not expressible in first-order logic.

The reason why we achieve decidability of the fully-uniform strategy problem for the class of rational relations is that most of the complexity of this problem is on the vertical axis, with branching quantifiers, temporal fixpoints and existence of a strategy. On the horizontal axis, our logic $\mathcal{L}_{\rightsquigarrow}$ can only quantify universally or existentially over immediate neighbors, like basic modal logic. Moreover, the fact that the semantics of the full quantifier is, unlike for the strict quantifier, independent of the strategy, allows us to treat both axis separately.

A related result concerning rational graphs is established by Bekker and Goranko (2007). They establish that model checking the basic modal logic **K** with forward and

backward modalities on rational graphs is decidable, and they conjecture that it is nonelementary.

A direction that we believe deserves investigation concerns information set automata. Because they allow for the computation of information sets, they may provide a way to define a general powerset construction for games with imperfect information and some subclass of rational relations. Of course this cannot work for arbitrary rational relations, as two-player games with imperfect information and safety objectives are already undecidable for some rational relations (see Remark 7). The two sorts of relations (apart from players with bounded memory) for which a powerset construction is known are synchronous and asynchronous perfect-recall (Reif, 1984; Chatterjee et al., 2006; Puchala, 2010). Identifying a class of relations that strictly contains these relations and for which a powerset construction exists would be interesting as it would give insights on what characteristics of players make these games decidable. This may also yield results concerning games with imperfect recall but unbounded memory, the study of which has recently received renewed attention (Berwanger et al., 2012).

This chapter terminates our study of uniform properties in $\mathcal{L}_{\rightsquigarrow}$. In Chapter 4 we focused on the strict quantifier, and in this chapter we studied the full quantifier. We could combine our techniques to allow for both kinds of quantifiers in a same uniformity property, but as our framework only allows for one relation we could only get a decidability result for recognizable relations. We could therefore not exploit the fact that full quantifiers can be handled for any rational relation. Thus we only combine strict and full quantifiers in the next chapter, in which we first generalize our framework to several relations.

Chapter 6

Generalization to several relations

In the previous chapters we only considered one relation between the nodes of trees. However in the context of multi-agent systems for example, it is very natural to consider several relations, representing the observational power of each agent (Parikh and Ramanujam, 1985; Fagin et al., 1995). It seems therefore relevant to extend the language $\mathcal{L}_{\sim}$ in order to allow for several different relations between nodes of trees, and associated quantifiers. In this chapter we extend the notion of uniform strategy to uniformity constraints involving several relations, and we study how results from Chapter 4 and Chapter 5 concerning the synthesis of uniform strategies generalize to this setting. Note that we only extend the logic, not the game model. We still consider two-player games, but the uniformity properties may involve several different relations between plays. However we informally describe in Section 6.5 how the distributed strategy problem in concurrent games with imperfect information can be reduced to a uniform strategy problem in two-player arenas.

We establish that, somehow surprisingly, all the results concerning decidability and complexity are unchanged, except for the case of fully-uniform strategies with K45NM relations, which is still decidable but no longer elementary when several relations are involved. In a second time, we describe how the extended setting allows us to establish a relevant decidability result for a class of uniformity constraints that can use both strict and full quantifiers in the same formula. Finally, we show how our techniques provide a new unified proof for a number of results from the literature concerning the model-checking of epistemic temporal logics. In fact we obtain a very general result concerning the model-checking problem of such logics for a wide class of epistemic relations.

In this chapter, for each uniform strategy problem that we prove decidable, the associated synthesis problem can also be solved with the same time complexity. The justification is alike the case of one relation (see Page 59 for strictly-uniform strategy synthesis and Page 79 for fully-uniform strategy synthesis). This concerns Theorems 12, 13, 14, 15 and 16.

Note that for no relation, all the logics considered collapse to CTL^*, and all the problems that we consider are 2-EXPTIME-complete. We therefore generalize $\mathcal{L}_{\sim}$ to n relations, where n is greater than 1.

6.1 Extending the language

We start by extending the syntax and semantics of $\mathcal{L}_{\rightsquigarrow}$ to handle n relations. For $n \in \mathbb{N}^*$, we call $n\mathcal{L}_{\rightsquigarrow}$ the extended language for n relations, the syntax of which is as follows:

$$\begin{array}{ll} \text{State formulas:} & \varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{A}\psi \mid \boxminus_i \varphi \mid \boxbox_i \varphi \\ \text{Path formulas:} & \psi ::= \varphi \mid \neg\psi \mid \psi \vee \psi \mid \mathbf{X}\psi \mid \psi \mathbf{U} \psi, \end{array}$$

where $p \in AP$ and $i \in \{1, \dots, n\}$.

The models now include one relation $\rightsquigarrow_i$ for each pair of quantifiers $\boxminus_i, \boxbox_i$. Given a family of relations $\{\rightsquigarrow_i\}_{1 \leq i \leq n}$, a 2^{AP}-labelled forest $\mathcal{U}$ (the universe), a 2^{AP}-labelled tree t, a node $x \in t$ and a state formula $\varphi \in n\mathcal{L}_{\rightsquigarrow}$, $\{\rightsquigarrow_i\}_{1 \leq i \leq n}, \mathcal{U}, t, x \models \varphi$ means that φ holds at node x of t in universe $\mathcal{U}$ and with relations $\{\rightsquigarrow_i\}_{1 \leq i \leq n}$. Again, we shall omit the relations and the universe when they are understood from the context. The case of path formulas is similar, replacing a node $x \in t$ with a branch $\lambda \in Branches(t)$. The semantics is unchanged, except for formulas of the form $\boxminus_i \varphi$ or $\boxbox_i \varphi$, for which it naturally becomes:

- $t, x \models \boxminus_i \varphi$ if for all $y \in t$ such that $x \rightsquigarrow_i y$, $t, y \models \varphi$
- $t, x \models \boxbox_i \varphi$ if for all $y \in \mathcal{U}$ such that $x \rightsquigarrow_i y$, $\mathcal{U}_y, y \models \varphi$

The definition of uniform strategies straightforwardly extends to this language. Let $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$ be a finite 2^{AP}-labelled game arena for some finite set AP. Let us note $\Sigma = 2^{AP}$, and let $\{\rightsquigarrow_i\}_{1 \leq i \leq n}$ be a family of binary relations over Σ^*. Finally, let φ be an $n\mathcal{L}_{\rightsquigarrow}$ formula. The universe $\mathcal{U}$ is still the (Σ, V)-forest of all paths in the arena starting from a position in V_ι: $\mathcal{U} = Paths_*(V_\iota)$.

Definition 36. A strategy σ is $(\{\rightsquigarrow_i\}_{1 \leq i \leq n}, \varphi)$-*uniform* if the strategy tree of σ satisfies φ, *i.e.* $t_\sigma \models \varphi$.

We also define the sublanguages $\mathcal{S}n\mathcal{L}_{\rightsquigarrow}$ and $\mathcal{F}n\mathcal{L}_{\rightsquigarrow}$ that generalize respectively $\mathcal{S}\mathcal{L}_{\rightsquigarrow}$ and $\mathcal{F}\mathcal{L}_{\rightsquigarrow}$.

Definition 37. Let $\mathcal{S}n\mathcal{L}_{\rightsquigarrow}$ and $\mathcal{F}n\mathcal{L}_{\rightsquigarrow}$ be the sublanguages of $n\mathcal{L}_{\rightsquigarrow}$ that use only one kind of quantifier: respectively, $\{\boxminus_i\}_{1 \leq i \leq n}$ quantifiers and $\{\boxbox_i\}_{1 \leq i \leq n}$ quantifiers.

We now study how our results on the synthesis of uniform strategies generalize to the case of n relations.

6.2 Strictly-uniform strategies

As we started with the study of strictly-uniform strategies in the one relation case, so do we now in the extended setting. First of all, the undecidability of the strictly uniform strategy problem for rational relations is straightforwardly transferred.

Definition 38.

$$n\mathsf{SUS} := \left\{ (\mathcal{G}, \{T_i\}_{1 \leq i \leq n}, \Phi) \;\middle|\; \begin{array}{c} \mathcal{G} \text{ is a finite } 2^{AP}\text{-labelled arena,} \\ \text{each } T_i \text{ is a transducer over } 2^{AP}, \\ \Phi \in \mathcal{S}n\mathcal{L}_{\rightsquigarrow}, \text{ and} \\ \text{there exists a } (\{[T_i]\}_{1 \leq i \leq n}, \Phi)\text{-uniform strategy} \\ \text{for Player 1 in } \mathcal{G}. \end{array} \right\}$$

Theorem 11. *n*SUS *is undecidable.*

However, we prove that the decidability of SUS restricted to recognizable relations also extends to the case of several relations. To do so, we first adapt our jumping automata of Section 4.2.

6.2.1 Extended jumping tree automata

Let Υ be a tree alphabet and Σ be a labelling alphabet. Jumping tree automata (JTA) over (Σ, Υ)-tree naturally extend to operate with several relations. We equip them with n binary relations over Σ^*, and we allow transition functions to use transition directions $\diamondsuit_i$ and $\boxdot_i$ for any $i \in \{1, \dots, n\}$. We let $nDir_{\rightsquigarrow} = Dir_A \cup \bigcup_{i=1}^{n} \{\diamondsuit_i, \boxdot_i\}$ denote the set of transition directions for JTA with n relations. Recall that Dir_A is the set of transition directions for alternating tree automata, which is normally $Dir_A = \Upsilon$. Once again for clarity of presentation, and because it is (most of the time) sufficient for our needs, we consider instead the abstract directions $Dir_A = \{\Diamond, \Box\}$. As for JTA with one relation, all our results also hold in the case where $Dir_A = \Upsilon$.

Definition 39. A $nDir_{\rightsquigarrow}$-automaton equipped with a family of relations $\{\rightsquigarrow_i\}_{1 \leq i \leq n}$ over Σ^* is a *multi jumping tree automaton*, or simply jumping tree automaton. [1]

Like for the case of one relation, JTA equipped with n relations capture the semantics of our language with n strict quantifiers, and they capture the strictly-uniform strategy problem.

Proposition 21. *Let φ be an $\mathcal{S}n\mathcal{L}_{\rightsquigarrow}$ formula, and let $\{\rightsquigarrow_i\}_{1 \leq i \leq n}$ be a family of relations over $(2^{AP})^*$. There exists a jumping tree automaton $\mathcal{A}_\varphi$ over alphabet 2^{AP} equipped with relations $\{\rightsquigarrow_i\}_{1 \leq i \leq n}$, with two colours and of size $2^{O(|\varphi|)}$ such that $t \in \mathcal{L}(\mathcal{A}_\varphi)$ if, and only if, $t \models \varphi$.*

Proof. Similar to Proposition 8, page 51. □

Proposition 22. *Let $(\mathcal{G}, \{T_i\}_{1 \leq i \leq n}, \Phi)$ be an instance of n*SUS*. There is a jumping tree automaton $\mathcal{A}$ equipped with relations $\{[T_i]\}_{1 \leq i \leq n}$ such that σ is a $(\{[T_i]\}_{1 \leq i \leq n}, \Phi)$-uniform strategy if, and only if, $t_\sigma \in \mathcal{L}(\mathcal{A})$. Moreover, $\mathcal{A}$ can be chosen of size $|\mathcal{A}| = |\mathcal{G}|^2 + 2^{O(|\Phi|)}$ and with only two colours.*

Proof. Similar to Proposition 9, page 53. □

This result tells us that, like in the case of a single relation, if for some class of relations the emptiness problem for jumping tree automata is decidable then the strictly-uniform strategy problem for this class of relation is also decidable. Because nSUS is undecidable (Theorem 11), the following corollary is immediate.

Corollary 6. *The emptiness problem for jumping tree automata equipped with regular equivalence relations is undecidable, hence it is also undecidable for rational relations.*

1. See Definition 6 for the definition of *Dir*-tree automata.

6.2.2 Decidable case

Even though the emptiness of JTA is undecidable for arbitrary rational relations, we show that, as for the case of a single relation, restraining to recognizable relations yields decidability via two-way tree automata. Indeed, when all the relations that equip a jumping automaton are recognizable, one can build an equivalent two-way tree automaton of linear size, the emptiness of which is decidable in exponential time. Recall that if a relation $\rightsquigarrow$ is recognizable, $\mathcal{B}_{\rightsquigarrow}$ denotes the minimal deterministic word automaton that recognizes it (see Fact 4, page 27).

Proposition 23. *If $\mathcal{A}$ is a jumping tree automaton equipped with n recognizable relations $\{\rightsquigarrow_i\}_{1\leq i\leq n}$ and using l colours, then there is a two-way tree automaton $\widehat{\mathcal{A}}$ using $O(l)$ colours and of size $O(|\mathcal{A}| \cdot \sum_{i=1}^{n} |\mathcal{B}_{\rightsquigarrow_i}|)$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\widehat{\mathcal{A}})$.*

Proof. The proof is similar to Proposition 11, page 55. The only detail is that when the two-way automaton has to simulate a jump for relation $\rightsquigarrow_i$, it triggers the associated recognizer $\mathcal{B}_i$, and it stops executing it when the jump mode ends. The two-way automaton never needs to execute several $\mathcal{B}_i$ at the same time, hence the size of the two-way automaton. □

We now establish two corollaries. The first one, Corollary 7, is a direct consequence of Proposition 23 and Theorem 4; the second one, Corollary 8, follows from the first one together with Proposition 2. Let $\mathcal{A}$ be a jumping tree automaton over alphabet Σ equipped with a family of recognizable relations $\{\rightsquigarrow_i\}_{1\leq i\leq n}$ and using l colours. Let m_a (resp. m_i) be the number of states in $\mathcal{A}$ (resp. $\mathcal{B}_{\rightsquigarrow_i}$), and let $m = m_a \cdot \sum_{i=1}^{n} m_i \cdot l$.

Corollary 7. *One can build a nondeterministic tree automaton $\mathcal{A}'$ with $2^{m \log(m)}$ states and $O(m)$ colours such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

Corollary 8. *The emptiness problem for $\mathcal{A}$ can be decided in time $|\Sigma|^{O(m)} \cdot 2^{O(d\, m^2 \log(m))}$, where d is the maximal arity of trees ($d = |\Upsilon|$ for Υ-trees).*

From this we obtain that the strictly-uniform strategy problem with multiple relations is decidable if all the relations are recognizable. In addition, the complexity is not higher than in the case of a single relation.

Definition 40.

$$n\mathsf{SUS}_{\mathrm{Rec}} := \left\{ (\mathcal{G}, \{\mathcal{B}_{\rightsquigarrow_i}\}_{1\leq i\leq n}, \Phi) \;\middle|\; \begin{array}{c} \mathcal{G} \text{ is a finite } 2^{AP}\text{-labelled arena,} \\ \text{each } \rightsquigarrow_i \text{ is a recognizable relation on } (2^{AP})^*, \\ \Phi \in \mathcal{S}n\mathcal{L}_{\rightsquigarrow}, \text{ and} \\ \text{there exists a } (\{\rightsquigarrow_i\}_{1\leq i\leq n}, \Phi)\text{-uniform strategy} \\ \text{for Player 1 in } \mathcal{G}. \end{array} \right\}$$

The size of an instance $(\mathcal{G}, \{\mathcal{B}_{\rightsquigarrow_i}\}_{1\leq i\leq n}, \Phi)$ of $\mathsf{SUS}_{\mathrm{Rec}}$ is the sum of the sizes of its components, plus the number of atomic propositions: $|(\mathcal{G}, \{\mathcal{B}_{\rightsquigarrow_i}\}_{1\leq i\leq n}, \Phi)| = |\mathcal{G}| + \sum_{i=1}^{n} |\mathcal{B}_{\rightsquigarrow_i}| + |\Phi| + |AP|$.

Proposition 24. *Let $(\mathcal{G}, \{\mathcal{B}_{\rightsquigarrow_i}\}_{1\leq i\leq n}, \Phi)$ be an instance of $n\mathsf{SUS}_{\mathrm{Rec}}$, and let $m_b = \sum_{i=1}^{n} m_i$, where m_i is the number of states in $\mathcal{B}_{\rightsquigarrow_i}$. Deciding whether $(\mathcal{G}, \{\mathcal{B}_{\rightsquigarrow_i}\}_{1\leq i\leq n}, \Phi)$ is a positive instance can be done in time $(|\mathcal{G}|^d \cdot 2^{|AP|+d\, m_b \log(m_b)})^{2^{O(|\Phi|)}}$, where d is the maximum branching degree in $|\mathcal{G}|$.*

Proof. The proof is similar to Proposition 12, page 58. □

This result establishes that the strictly-uniform strategy problem for recognizable relations has essentially the same complexity whether one or several relations are allowed.

Theorem 12. $n\mathsf{SUS}_{\mathrm{Rec}}$ *is* 2-EXPTIME*-complete.*

Proof. The lower bound is inherited from the case of one relation, and Proposition 24 provides the upper bound. □

We have established that for strictly-uniform strategies, our decidability and complexity results coincide for the single and the multi relations cases. We now turn to fully-uniform strategies.

6.3 Fully-uniform strategies

In this section we first prove that the fully-uniform strategy problem for n relations remains decidable. However, while for the class of rational relations, allowing for several relations leaves the complexity of the problem unchanged (nonelementary), it is not so for the class of K45NM relations. Indeed, while the fully-uniform strategy problem is 2-EXPTIME-complete for a single K45NM relation, alternation between full quantifiers for different relations makes the problem nonelementary for n relations. However the complexity remains lower than for arbitrary rational relations: it depends on the *alternation* between full quantifiers for different relations, rather than the total nesting depth of full quantifiers.

6.3.1 Rational relations

First, we generalize the notion of $\boxbslash$-depth. The $\boxbslash$-depth of an $\mathcal{F}n\mathcal{L}_{\rightsquigarrow}$ formula φ, still written $d(\varphi)$, is the maximum number of nested $\boxbslash_i$ quantifiers in φ, regardless of the value of i. Formally, $d(\varphi)$ is defined inductively as follows:

$$d(p) = 0 \qquad d(\neg\varphi) = d(\varphi) \qquad d(\varphi \vee \varphi') = \max(d(\varphi), d(\varphi'))$$
$$d(\mathbf{A}\psi) = d(\psi) \qquad d(\mathbf{X}\psi) = d(\psi) \qquad d(\psi \mathbf{U} \psi') = \max(d(\psi), d(\psi'))$$
$$d(\boxbslash_i \varphi) = 1 + d(\varphi)$$

We define the following decision problems.

Definition 41. For each $k \in \mathbb{N}$, we let

$$n\mathsf{FUS}_k := \left\{ (\mathcal{G}, \{T_i\}_{1\leq i\leq n}, \Phi) \;\middle|\; \begin{array}{c} \mathcal{G} \text{ is a finite } 2^{AP}\text{-labelled arena,} \\ \text{each } T_i \text{ is a transducer over } 2^{AP}, \\ \Phi \in \mathcal{F}n\mathcal{L}_{\rightsquigarrow}, d(\Phi) \leq k, \text{ and} \\ \text{there is a } (\{[T_i]\}_{1\leq i\leq n}, \Phi)\text{-uniform strategy} \\ \text{for Player 1 in } \mathcal{G}. \end{array} \right\}$$

and the *fully-uniform strategy problem* is

$$n\mathsf{FUS} := \bigcup_{k\in\mathbb{N}} n\mathsf{FUS}_k.$$

As usual, we let the size of an instance be the sum of the sizes of its components plus the number of atomic propositions: $|(\mathcal{G}, \{T_i\}_{1\leq i\leq n}, \Phi)| = |\mathcal{G}| + \sum_{i=1}^{n} |T_i| + |\Phi| + |AP|$.

Theorem 13. *$n\mathsf{FUS}_k$ is k-*Exptime*-complete if $k \geq 2$, 2-*Exptime*-complete otherwise.*

Proof. The lower bounds are directly inherited from FUS_k. For the upper bounds, let $(\mathcal{G}, \{T_i\}_{1\leq i\leq n}, \Phi)$ be an instance of $n\mathsf{FUS}_k$ for some $k > 0$, with $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$. We explain informally how to adapt the construction described in Section 5.3.1. From each transducer T_i (recognizing relation $\rightsquigarrow_i$), build the associated information set automaton $\mathcal{A}^{T_i}$, and synchronize the arena $\mathcal{G}$ with $\mathcal{A}^{T_1}, \ldots, \mathcal{A}^{T_n}$. This gives, by synchronous product, a powerset arena $\widehat{\mathcal{G}}$ of size $2^{(|\mathcal{G}|+|T_1|+\ldots+|T_n|)^{O(1)}}$ in which each innermost subformula of the form $\boxbslash_i\varphi$ can be evaluated positionally, using the information computed by $\mathcal{A}^{T_i}$. A slight modification of Algorithm 1 marks the positions of $\widehat{\mathcal{G}}$ with the fresh propositions $p_{\boxbslash_i\varphi}$ where $\boxbslash_i\varphi$ holds (for $\boxbslash_i\varphi \in \mathit{Sub}(\Phi)$). This provides an exponential-size instance of $n\mathsf{FUS}_{k-1}$, which can be proven equivalent [2] to the original one by an easy adaptation of Proposition 16 (page 73).

The upper bounds are then established by induction on k, like in Proposition 17. □

Corollary 9. *$n\mathsf{FUS}$ is nonelementary.*

6.3.2 K45NM relations

We show that the complexity of the fully-uniform strategy problem for K45NM relations rises from 2-Exptime-complete for one relation to nonlementary for n relations. However, the details show that the complexity remains better than for the case of arbitrary rational relations. Recall that in the case of one relation, the properties of K45NM relations enable us to eliminate all $\boxbslash$ quantifiers by a single powerset construction, while for rational relations one additional powerset construction is needed for each nesting of $\boxbslash$ quantifier. In the case of several relations, the properties of K45NM relations still allow us to eliminate in one shot all sequences of nested $\boxbslash_i$ quantifiers, as long as the quantifiers concern the same relation. Nonetheless, each *alternation* between different quantifiers requires a new powerset construction.

We inductively define the *alternation depth* of a formula $\varphi \in \mathcal{F}n\mathcal{L}_{\rightsquigarrow}$, noted $ad(\varphi)$, as follows:

$$ad(p) = 0 \qquad ad(\neg\varphi) = ad(\varphi) \quad ad(\varphi \vee \varphi') = \max(ad(\varphi), ad(\varphi'))$$
$$ad(\mathbf{A}\psi) = ad(\psi) \quad ad(\mathbf{X}\psi) = ad(\psi) \quad ad(\psi\mathbf{U}\psi') = \max(ad(\psi), ad(\psi'))$$
$$ad(\boxbslash_i\varphi) = 1 + \max\{ad(\boxbslash_j\varphi') \mid \boxbslash_j\varphi' \in \mathit{Sub}(\varphi) \text{ and } i \neq j\}\ [3]$$

For a detailed study of the fully-uniform strategy problem with several K45NM relations, we bound the alternation depth of formulas.

2. In the sense that the first instance is a positive instance if, and only if, the second also is.
3. Classically, $\max(\emptyset) = 0$.

Definition 42. For each $h \in \mathbb{N}$, we let:

$$n\mathsf{FUS}^h_{\mathrm{K45NM}} := \left\{ (\mathcal{G}, \{T_i\}_{1\leq i\leq n}, \Phi) \;\middle|\; \begin{array}{c} \mathcal{G} \text{ is a finite } 2^{AP}\text{-labelled arena,} \\ \text{each } T_i \text{ is a transducer over } 2^{AP} \\ \text{such that } [T_i] \in \mathrm{K45NM}, \\ \Phi \in \mathcal{F}n\mathcal{L}_{\rightsquigarrow},\ ad(\Phi) \leq h, \text{ and} \\ \text{there is a } (\{[T_i]\}_{1\leq i\leq n}, \Phi)\text{-uniform strategy} \\ \text{for Player 1 in } \mathcal{G}. \end{array} \right\}$$

and

$$n\mathsf{FUS}_{\mathrm{K45NM}} := \bigcup_{h\in\mathbb{N}} n\mathsf{FUS}^h_{\mathrm{K45NM}}.$$

We have proven that for $n = 1$, the fully-uniform strategy problem is 2-Exptime-complete (Theorem 10). For $n > 1$ we establish the following result.

Theorem 14. *For $n > 1$, $n\mathsf{FUS}^h_{\mathrm{K45NM}}$ is h-Exptime-complete if $h \geq 2$, otherwise it is 2-Exptime-complete.*

Proof. Let $n > 1$, and let $(\mathcal{G}, \{T_i\}_{1\leq i\leq n}, \Phi)$ be an instance of $n\mathsf{FUS}^h_{\mathrm{K45NM}}$ for some $h > 0$. The procedure for $\mathsf{FUS}_{\mathrm{K45NM}}$ (Section 5.3.2) can be applied iteratively to remove alternation-free subformulas of the form $\boxminus_i\varphi$. Roughly speaking, using the fact that for each relation $\rightsquigarrow_i$, two related paths take the information set automaton $\mathcal{A}^{T_i}$ in two information-set bisimilar states, one can recursively evaluate on the first powerset arena $\widetilde{\mathcal{G}}$ all innermost subformulas of the form $\boxminus_i\varphi$, like in Algorithm 2 (page 77), until alternations are reached.

For example, for a formula $\varphi = \boxminus_1\boxminus_2\boxminus_2 q \wedge \boxminus_1\boxminus_2\boxminus_1 q$, the formula we obtain after this procedure is

$$\widetilde{\varphi} = \boxminus_1 p_{\boxminus_2 p_{\boxminus_2 q}} \wedge \boxminus_1\boxminus_2 p_{\boxminus_1 q},$$

where $p_{\boxminus_1 q}$, $p_{\boxminus_2 q}$ and $p_{\boxminus_2 p_{\boxminus_2 q}}$ are fresh atomic propositions.

This yields an equivalent instance of $n\mathsf{FUS}^{h-1}_{\mathrm{K45NM}}$, the size of which is exponential in $|(\mathcal{G}, \{T_i\}_{1\leq i\leq n}, \Phi)|$. An induction on h like in Proposition 17 (page 75) gives the desired upper bounds.

For the lower bounds, the construction presented for FUS in Proposition 30 of the appendix (Page 130) can be adapted. The behaviour of the transducer T built in the proof of Lemma 27 (Page 132) shows that the relation it recognizes is not in K45NM. First, we describe how to obtain K45NM relations by alternating between two transducers that each recognize a transitive and Euclidean relation. To do so, we make each transducer label its outputs with a special atomic proposition that identifies the transducer that produced the output. Say that the first transducer, T_0, uses p_{T_0} and T_1 uses p_{T_1}. First, observe that an input ρ labelled by some p_{T_i} has necessarily been output by T_i on some input ρ', and furthermore a transducer reading ρ can "know" exactly what was this ρ'. Indeed, the contents of ρ and ρ' are the same, and we can add finitely many atomic propositions to mark on an output what was the kind of the input (there are but a finite number of different kinds of plays). Therefore we can let T_i behave on a p_{T_i}-tagged play ρ as it would

have behaved on the input ρ' that originated ρ. This ensures that on a given input of T_i, all the outputs are related together, so that $[T_i]$ is transitive and Euclidean. On a play that is tagged by neither p_{T_0} nor p_{T_1}, T_i just behaves normally and tags the outputs with p_{T_i}. On plays tagged with $p_{T_{1-i}}$, T_i does the same, but in addition it removes the tag $p_{T_{1-i}}$. In the formulas built in the proof, we alternate between full quantifiers for each transducer, such that each T_i always receives as input a play that is not tagged with p_{T_i}, and it therefore behaves normally.

Now, making the relations verify the No Miracles property can be done by adding loops on accepting states of the transducers with labels a/a for each a in the alphabet but, as a side effect, insignificant related plays are added. This issue can however be dealt with by adding two new atomic propositions, $valid_0$ and $valid_1$, used by transducers to signal which related plays are relevant and which ones are added to achieve the No Miracles property. More precisely, when T_i reads a partial play, in addition to its normal behaviour (described above) it also marks each position in the output with $valid_i$, except in loop transitions added for the No Miracles property, that still just copy the input. Making sure that transducer T_i never takes as input a word already marked with $valid_i$ ensures that the relevant outputs of T_i and only them are marked with $valid_i$. This is achieved by letting each transducer erase the marker of the other, by relativising the $\boxbslash_i$-quantifications to $valid_i$-marked plays, and specifying in the main formula that the plays in the strategy must be originally unmarked (see Lemma 27). □

Corollary 10. *$n\mathsf{FUS}_{\mathrm{K45NM}}$ is nonelementary.*

In the next section we show how techniques for strict and full uniformity can be combined.

6.4 Mixing strict and full quantifiers

The fact that $n\mathcal{L}_{\rightsquigarrow}$ allows for quantifiers over different relations (when $n > 1$) makes it possible to define a relevant class of uniformity constraints that combine strict and full quantifiers. To give the idea, consider an $\mathcal{L}_{\rightsquigarrow}$ formula φ that contains no strict quantifier in the scope of a full quantifier. This means that if $\boxbslash_i\varphi' \in Sub(\varphi)$, then there is no formula of the form $\bslash_j\varphi''$ in $Sub(\varphi')$, for any j. If the relations attached to the full quantifiers in φ are rational, then we can iterate powerset constructions and subformula elimination of Section 5.3 to remove all full quantifiers. Recall that there is a bijection between plays in an arena and plays in its powerset construction. Because this bijection preserves relations, the truth of strict quantifiers is also preserved between a tree of the original arena and a tree of the powerset arena. Therefore, eliminating the full quantifiers yields an equivalent instance of $n\mathsf{SUS}$. This instance can in turn be solved using jumping tree automata, provided the relations attached to the strict quantifiers are recognizable.

We let $\mathcal{SF}n\mathcal{L}_{\rightsquigarrow}$ be the set of formulas in $n\mathcal{L}_{\rightsquigarrow}$ such that there is no strict quantifier in the scope of a full quantifier. For example, $\mathbf{AG}\boxbslash_2 q \in \mathcal{SF}n\mathcal{L}_{\rightsquigarrow}$, $\bslash_1\mathbf{EF}\boxbslash_2\boxbslash_1 p \in \mathcal{SF}n\mathcal{L}_{\rightsquigarrow}$, but $\boxbslash_1\mathbf{AX}\bslash_2 p \notin \mathcal{SF}n\mathcal{L}_{\rightsquigarrow}$. We start with the general case, where the relations for the full quantifiers can be arbitrary rational relations.

6.4.1 Rational relations

The $\boxminus$-depth, originally defined for $\mathcal{FL}_{\rightsquigarrow}$ formulas and extended to $\mathcal{FnL}_{\rightsquigarrow}$ formulas in Section 6.3.1, is now extended to $\mathcal{SFnL}_{\rightsquigarrow}$ formulas (and in fact to arbitrary $n\mathcal{L}_{\rightsquigarrow}$ formulas). The $\boxminus$-depth of an $n\mathcal{L}_{\rightsquigarrow}$ formula φ, still written $d(\varphi)$, simply counts the nesting of full quantifiers and ignores strict quantifiers. It is defined inductively on formulas as for $\mathcal{FnL}_{\rightsquigarrow}$ formulas, the new inductive case being: $d(\boxminus_i\varphi) = d(\varphi)$.

Definition 43. For each $k \in \mathbb{N}$, we let:

$$n\mathsf{SFUS}_k := \left\{ \begin{pmatrix} \mathcal{G}, \\ \{\mathcal{B}_{\rightsquigarrow_i}\}_{1\leq i\leq m}, \\ \{T_i\}_{m<i\leq n}, \\ \Phi \end{pmatrix} \;\middle|\; \begin{array}{c} \mathcal{G} \text{ is a finite } 2^{AP}\text{-labelled arena, } 0 \leq m \leq n, \\ \text{for } 1 \leq i \leq m,\ \rightsquigarrow_i \text{ is a recognizable relation,} \\ \text{for } m < i \leq n, T_i \text{ is a transducer over } 2^{AP}, \\ \Phi \in \mathcal{SFnL}_{\rightsquigarrow}, d(\Phi) \leq k, \\ \text{if } \boxminus_i\varphi \in \mathit{Sub}(\Phi), \text{ then } 1 \leq i \leq m, \\ \text{if } \boxplus_i\varphi \in \mathit{Sub}(\Phi), \text{ then } m < i \leq n, \text{ and} \\ \text{letting } \rightsquigarrow_i = [T_i] \text{ for } m < i \leq n, \\ \text{there is a } (\{\rightsquigarrow_i\}_{1\leq i\leq n}, \Phi)\text{-uniform strategy} \\ \text{for Player 1 in } \mathcal{G}. \end{array} \right\}$$

and

$$n\mathsf{SFUS} := \bigcup_{k\in\mathbb{N}} n\mathsf{SFUS}_k.$$

Remark 11. An instance $(\mathcal{G}, \{\mathcal{B}_{\rightsquigarrow_i}\}_{1\leq i\leq m}, \{T_i\}_{m<i\leq n}, \Phi)$ of $n\mathsf{SFUS}_k$ where $m = 0$ is an instance of $n\mathsf{FUS}_k$. On the opposite, if $m = n$ then it is an instance of $n\mathsf{SUS}$. Also, if $k = 0$, there is no full quantifier in Φ, therefore it is also an instance of $n\mathsf{SUS}$.

Theorem 15. *$n\mathsf{SFUS}_k$ is k-*Exptime*-complete for $k \geq 2$, 2-*Exptime*-complete otherwise.*

Proof. The lower bounds are inherited from those of $n\mathsf{FUS}_k$.

For the upper bounds, let $(\mathcal{G}, \{\mathcal{B}_{\rightsquigarrow_i}\}_{1\leq i\leq m}, \{T_i\}_{m<i\leq n}, \Phi)$ be an instance of $n\mathsf{SFUS}_k$. If $k = 0$, then it is an instance of $n\mathsf{SUS}$ (see Remark 11) and can be solved in time doubly exponential (Theorem 12).

If $k > 0$, first build the powerset arena $\widehat{\mathcal{G}}$ of size $2^{(|\mathcal{G}|+|T_1|+\ldots+|T_n|)^{O(1)}}$, as done for $n\mathsf{FUS}_k$. Because there is no strict quantifier in the scope of a full quantifier, innermost formulas of the form $\boxplus_i\varphi$ can be evaluated and replaced by fresh atomic propositions. Compared to the case of $n\mathsf{FUS}$, here in addition we have to adapt each recognizer $\mathcal{B}_i$ to work on alphabet $2^{\widehat{AP}}$, where $\widehat{AP}$ is the set of atomic propositions used in the original instance augmented with the fresh atomic propositions. The automaton $\widehat{\mathcal{B}}_i$ works just like $\mathcal{B}_i$, by ignoring the fresh atomic propositions in the labellings it reads. Note that $\widehat{\mathcal{B}}_i$ has the same number of states as $\mathcal{B}_i$.

This procedure yields an equivalent instance of $n\mathsf{SFUS}_{k-1}$. One can check in particular that, by letting $\widehat{\boxminus_i\varphi} = \boxminus_i\widehat{\varphi}$, Lemma 13 (page 73) still holds in this context. The additional inductive case $\varphi = \boxminus_i\varphi'$ in the proof of Lemma 13 is routine, by using the canonical bijection between plays of $\mathcal{G}$ and plays of $\widehat{\mathcal{G}}$. Iterating the process k times removes all full quantifiers. However, all the strict quantifiers from the original formula Φ are unaffected

by this procedure. Therefore, while for $n\mathsf{FUS}_k$ the iterated elimination of full quantifiers yields an equivalent CTL^* game, here we obtain instead an equivalent strictly-uniform strategy problem. More precisely, the instance we obtain is an instance of $n\mathsf{SUS}_{\mathrm{Rec}}$, which is decidable by Theorem 12. Indeed, it is assumed that if $\boxminus_i\varphi \in \mathit{Sub}(\Phi)$ then $1 \leq i \leq m$, meaning that every strict quantifier $\boxminus_i$ in Φ corresponds to a recognizable relation (see Definition 43 of $n\mathsf{SFUS}_k$).

Let $(\widehat{\mathcal{G}}, \{\widehat{\mathcal{B}}_i\}_{1\leq i\leq m}, \widehat{\Phi})$ be the instance of $n\mathsf{SUS}_{\mathrm{Rec}}$ obtained by the procedure described above, and let $\widehat{AP}$ be its set of atomic propositions. The arena $\widehat{\mathcal{G}}$ is computed in time $\exp^k(|(\mathcal{G}, \{T_i\}_{m<i\leq n}, \Phi)|^{O(1)})$, which is also its size. Let $m_b = \sum_{i=1}^{n} m_i$, where m_i is the number of states in $\widehat{\mathcal{B}}_i$. Recall that $\widehat{\mathcal{B}}_i$ has a many states as $\mathcal{B}_i$. Also, let d be the maximum branching degree in $|\widehat{\mathcal{G}}|$. By Proposition 24, deciding whether $(\widehat{\mathcal{G}}, \{\widehat{\mathcal{B}}_i\}_{1\leq i\leq m}, \widehat{\Phi}) \in n\mathsf{SUS}_{\mathrm{Rec}}$ can be done in time $(|\widehat{\mathcal{G}}|^d \cdot 2^{|\widehat{AP}|+d\, m_b \log(m_b)})^{2^{O(|\Phi|)}} \leq (|\widehat{\mathcal{G}}|^d \cdot 2^{|AP|+|\Phi|+d\, m_b \log(m_b)})^{2^{O(|\Phi|)}}$, therefore deciding whether $(\mathcal{G}, \{\mathcal{B}_{\rightsquigarrow_i}\}_{1\leq i\leq m}, \{T_i\}_{m<i\leq n}, \Phi) \in n\mathsf{SFUS}_k$ can be done in time $(\exp^k(|(\mathcal{G}, \{T_i\}_{m<i\leq n}, \Phi)|^{O(1)})^d \cdot 2^{|AP|+|\Phi|+d\, m_b \log(m_b)})^{2^{O(|\Phi|)}}$.

To obtain the upper bounds it remains to observe that, as mentioned in the proof of Proposition 17 (page 75), the definition of our powerset construction ensures that the maximum branching degrees of the successive powerset arenas remains unchanged, such that $d \leq |\mathcal{G}|$. □

Corollary 11. *$n\mathsf{SFUS}$ is nonelementary.*

6.4.2 K45NM relations

We also consider the particular case where all the relations involved in full quantifiers are in K45NM. As for $n\mathsf{FUS}_{\mathrm{K45NM}}$, the height of the tower of exponentials in the time complexity of solving the problem drops from the total nesting depth of full quantifiers to the alternation depth.

For this, we extend the notion of alternation depth to arbitrary $n\mathcal{L}_{\rightsquigarrow}$ formulas. For a formula $\varphi \in n\mathcal{L}_{\rightsquigarrow}$, its alternation depth is defined inductively as in Section 6.3.2. The additional inductive case is as follows: $\mathit{ad}(\boxminus_i\varphi) = \mathit{ad}(\varphi)$.

The problem that we consider is the following restriction of $n\mathsf{SFUS}$.

Definition 44. For each $h \in \mathbb{N}$, we let:

$$n\mathsf{SFUS}^h_{\mathrm{K45NM}} := \left\{ \left(\begin{array}{c} \mathcal{G}, \\ \{\mathcal{B}_{\rightsquigarrow_i}\}_{1\leq i\leq m}, \\ \{T_i\}_{m<i\leq n}, \\ \Phi \end{array} \right) \;\middle|\; \begin{array}{c} \mathcal{G} \text{ is a finite } 2^{AP}\text{-labelled arena, } 0 \leq m \leq n, \\ \text{for } 1 \leq i \leq m,\ \rightsquigarrow_i \text{ is a recognizable relation,} \\ \text{for } m < i \leq n, T_i \text{ is a transducer over } 2^{AP} \\ \text{such that } [T_i] \in \mathrm{K45NM}, \\ \Phi \in \mathcal{SF}n\mathcal{L}_{\rightsquigarrow}, \mathit{ad}(\Phi) \leq h, \\ \text{if } \boxminus_i\varphi \in \mathit{Sub}(\Phi), \text{ then } 1 \leq i \leq m, \\ \text{if } \blacksquare_i\varphi \in \mathit{Sub}(\Phi), \text{ then } m < i \leq n, \text{ and} \\ \text{letting } \rightsquigarrow_i = [T_i] \text{ for } m < i \leq n, \\ \text{there is a } (\{\rightsquigarrow_i\}_{1\leq i\leq n}, \Phi)\text{-uniform strategy} \\ \text{for Player 1 in } \mathcal{G}. \end{array} \right\}$$

and

$$n\mathsf{SFUS}_{\mathrm{K45NM}} := \bigcup_{h\in\mathbb{N}} n\mathsf{SFUS}^h_{\mathrm{K45NM}}.$$

Remark 12. For $n = 1$, an instance $(\mathcal{G}, \{\mathcal{B}_{\sim_i}\}_{1\leq i\leq m}, \{T_i\}_{m<i\leq 1}, \Phi)$ of $n\mathsf{SFUS}^h_{\mathrm{K45NM}}$ is either an instance of $\mathsf{FUS}_{\mathrm{K45NM}}$ if $m = 0$ or an instance of $\mathsf{SUS}_{\mathrm{Rec}}$ if $m = 1$. In both cases the problem can be solved in doubly exponential time (Theorem 10, page 65 and Theorem 8, page 59 respectively). In particular, the alternation depth of Φ is never more than 1.

Theorem 16. *For $n = 1$, $n\mathsf{SFUS}^h_{\mathrm{K45NM}}$ is* 2-Exptime*-complete for all h. For $n > 1$, $n\mathsf{SFUS}^h_{\mathrm{K45NM}}$ is h-*Exptime*-complete for $h \geq 2$,* 2-Exptime*-complete otherwise.*

Proof. The lower bounds are inherited from those of $n\mathsf{FUS}^h_{\mathrm{K45NM}}$. For the upper bounds, the proof is similar to Theorem 15. The difference is that, as justified in the proof of Theorem 14, when the relations for the full quantifiers are in K45NM, a number of powerset constructions equal to the alternation depth of the input formula is sufficient to eliminate all full quantifiers and obtain an equivalent instance of $n\mathsf{SUS}_{\mathrm{Rec}}$. □

Corollary 12. *For $n > 1$, $n\mathsf{SFUS}_{\mathrm{K45NM}}$ is nonelementary. Otherwise it is* 2-Exptime*-complete.*

In the next section we discuss some impacts of the results established in this chapter, as well as connections with known results from the literature.

6.5 Application of our results and related work

In this section we discuss some known results related to the contributions of this chapter. We first point out two results from the literature on logics of knowledge and time that are direct corollaries of our results on uniform strategies. In a second time we discuss connections with the problem of distributed strategy synthesis with epistemic temporal objectives.

6.5.1 Model checking knowledge and time

Recall that LTLK_n (resp. CTLK_n) is LTL (resp. CTL) enriched with n knowledge operators $K_1, \ldots, K_n$. The models of these logics are usually interpreted systems, *i.e.* transition systems with valuations on states, and for each knowledge operator K_i, an equivalence relation $\sim_i$ on states of the system. Each relation is extended to sequences of states according to assumptions made on the memory of the agents (see, *e.g.* van der Meyden and Shilov (1999) for more detail). Almost all studies concern memoryless agents or synchronous perfect-recall agents. Note that, setting aside the strategic aspect, our labelled game arenas together with relations on plays can simulate labelled transition systems.

Theorem 17 (van der Meyden and Shilov (1999))**.** *Model-checking LTLK_n with synchronous perfect recall is decidable.*

Theorem 18 (Dima (2008))**.** *Model-checking CTLK_n with synchronous perfect recall is decidable.*

Because our base language is CTL^* and synchronous perfect-recall relations are in K45NM, both problems are easily reduced to $n\mathsf{FUS}_{\mathrm{K45NM}}$. Let $\mathcal{M}$ be an interpreted system and let φ a formula of either LTLK_n or CTLK_n. For each epistemic relation $\sim_i$, let $\simeq_i$ denote its classic extension to sequences of states according to the perfect recall synchronous semantics. We define a one-state transducer T_i which has a transition on a pair of states (s, s') whenever $s \sim_i s'$. Clearly, T_i recognizes $\simeq_i$. Being an equivalence relation, $\simeq_i$ is in K45, and one easily checks that it satisfies No Miracles. Then, one simply sees the model $\mathcal{M}$ as an arena whose positions all belong to Player 2. Player 1 has only one trivial strategy in this game, and all possible runs in the model are in the outcome. So φ is true in the model if and only if the only trivial strategy of Player 1 is $(\{[T_i]\}_{1\leq i\leq n}, \varphi)$-uniform (if $\varphi \in \mathsf{LTLK}_n$, take its universal quantification).

The algorithms given in van der Meyden and Shilov (1999) and Dima (2008) are in k-EXPSPACE for formulas of knowledge nesting depth k. Our results improve this upper bound: Theorem 14 gives an h-EXPTIME upper-bound, where h is the alternation depth of the formula instead of the total nesting of knowledge quantifiers. Note that for $h = 1$ we manage to obtain an EXPTIME upper-bound instead of the 2-EXPTIME one we have for $n\mathsf{FUS}^1_{\mathrm{K45NM}}$. The reason is that the automaton $\mathcal{A}_{\mathcal{G}}$ that recognizes the strategy trees of Player 1 in the above reduction is trivial as Player 1 never plays. In fact we can combine this automaton with the hesitant alternating automaton $\mathcal{A}_{\widehat{\Phi}}$ that accepts models of the final CTL^* formula, the automaton we obtain is still hesitant, and its emptiness can be tested in linear time (see Kupferman et al. (2000)). $\mathcal{A}_{\widehat{\Phi}}$ being of size exponential in $|\Phi|$, the result follows. In fact we have the following general result:

Theorem 19. *Model-checking $\mathsf{CTL}^*\mathsf{K}_n$ with rational epistemic relations on runs is in k-*EXPTIME*, where k is the modal depth of the formula. If, in addition, the epistemic relations verify transitivity, Euclideanity and No Miracles, then the problem is in h-*EXPTIME*, where h is the alternation depth of the formula.* K45NM *relations include,* e.g. *synchronous and asynchronous perfect-recall relations.*

Note that the same upper bounds for asynchronous perfect recall and distributed knowledge have also been proved by Aucher (2013).

6.5.2 Distributed strategy synthesis for epistemic temporal objectives

Uniform strategies also naturally capture the synthesis problem from knowledge-based specifications, as addressed in van der Meyden and Vardi (1998) or van der Meyden and Wilke (2005). We informally describe how this problem can be reduced to a uniform strategy problem.

Consider a concurrent game structure $\mathcal{G}$ where n players each have imperfect information and play against nature. Consider also a specification $\varphi \in \mathsf{LTLK}_n$ that the players aim at achieving. The problem is to decide whether there is a distributed strategy – one strategy for each player – such that the specification φ is enforced against all behaviours of nature. The imperfect information is classically defined by a set of observations O_i for each player i, and a mapping obs_i from positions of the game to O_i. Each player's observation is assumed to be synchronous perfect recall. Finally, each player plays by choosing actions from some private set Act_i of actions, and of course each player's strategy must be observation-based.

We reduce this problem to an instance of nSUS. To do so, we build a two-player arena $\mathcal{G}'$ where Player 1 simulates all the players but nature, which is represented by Player 2. First, as we do in Section 3.2 for two-player games with imperfect information, we put the players' actions inside newly created positions. For each position v and each tuple of actions $(a_1, \ldots, a_n)$, Player 1 can move from v to $(v, (a_1, \ldots, a_n))$. From there, nature can choose a next position v' according to the moves allowed in $\mathcal{G}$. Also, each position $(v, (a_1, \ldots, a_n))$ is labelled with atomic propositions $\{p_{a_1}, \ldots, p_{a_n}\}$, and positions where it is Player 1's turn to play (of the form v) are labelled with p_1. Because strategies must be observation based, for each player i we define, according to obs_i, the rational relation $\simeq_i$ over plays of $\mathcal{G}'$, that represents player i's observation. It remains to constrain Player 1 to play uniformly with respect to each player i. Recall from Section 3.2 the formula `SameAct` that characterizes observation-based strategies for one player. We define, for each i:

$$\texttt{SameAct}_i := \mathbf{AG}(p_1 \rightarrow \bigvee_{a_i \in \mathrm{Act}_i} \boxminus_i \mathbf{EX} p_{a_i}).$$

This formula means that Player 1 simulates player i's actions in a manner consistent with obs_i. Finally, observe that the knowledge operators in van der Meyden and Vardi (1998) or van der Meyden and Wilke (2005) have – to use the vocabulary of uniform strategies – strict semantics. Indeed, the system in which formulas are interpreted is restricted to behaviors that are induced by the synthesized strategy. Therefore, letting $\varphi' \in \mathcal{S}n\mathcal{L}_{\sim}$ be the formula φ where each knowledge modality K_i is replaced with the strict quantifier $\boxminus_i$, we have:

Proposition 25. *The distributed synthesis problem for φ in $\mathcal{G}$ has a solution if, and only if, there is a $(\{\simeq_i\}_{1 \leq i \leq n}, \varphi' \wedge \bigwedge_{i=1}^{n} \texttt{SameAct}_i)$-uniform strategy for Player 1 in $\mathcal{G}'$.*

van der Meyden and Vardi (1998) establish that the case of one agent with synchronous perfect recall is decidable; because perfect-recall relations are not recognizable relations, our contribution does not enable to obtain this result. However, this indicates that there should be a class of relations other than Rec for which the strictly-uniform strategy problem is decidable. On the other hand, when more than one agent are involved, the synthesis problem for LTLK_n specifications is undecidable; yet it is well-known that making additional assumptions on information flows yields decidable cases. For instance van der Meyden and Wilke (2005) establish that the problem is decidable in *broadcast environments*, where information is exchanged between players by means of public broadcasts only. This suggests that there should be relations not in Rec for which, under some assumptions on the connections between relations for each quantifier $\boxminus_i$, nSUS is decidable. It would be very interesting to identify for what class of relations and which constraints on their inter-connections this result holds.

Finally, consider the results of Section 6.4 where formulas can involve both strict and full quantifiers. This setting allows us to express in a single uniformity property that, for example, a strategy is observation-based (using the strict quantifier), and that it verifies some epistemic objective concerning the knowledge of players, who either know the strategy (strict quantifier) or ignore it (full quantifier). In Section 7.3.2 we discuss further this possibility to reason about the knowledge of agents with different abilities, and we propose an interpretation of the assumptions made to obtain decidability.

6.6 Conclusion

In this chapter we generalized our framework to manage several relations between plays in a game, instead of only one in the previous chapters. We established that our results from Chapters 4 and 5 still hold when the uniformity properties of strategies can involve quantifications for different relations over plays. The only difference concerns the case of fully-uniform strategies with K45NM relations, for which switching to several relations raises the complexity of the uniform strategy problem from 2-Exptime to nonelementary in the alternation depth of the formula. We then proved that techniques from Chapter 4 and 5 can be combined to solve the uniform strategy problem for a class of uniformity properties involving both strict and full quantifiers. Finally we gave two examples of frameworks studied in the literature that we can capture in our extended notion of uniform strategies.

The first one is the model checking of epistemic temporal logics. Because the knowledge modalities in this problem can be expressed with full quantifiers, our contribution provides an alternative, unified proof of decidability for these logics with synchronous and asynchronous perfect recall. In fact, we have the more general result that the model checking problem for these logics is decidable for any epistemic relations that are recognizable by finite state transducers, which is the case of synchronous and asynchronous perfect recall. In addition, as these relations are instances of K45NM relations, our contribution provides accurate upper bounds on the time complexity of these problems, in terms of alternation depth of knowledge modalities.

The second framework that we capture is the distributed strategy synthesis problem for LTLK_n specifications. We described how, using uniformity properties with several strict quantifiers, we can reduce the problem of solving a concurrent game with n players and imperfect information to solving a uniform strategy problem in a two-player game arena.

For players having perfect recall, the existence of distributed strategies in such games is undecidable, and indeed the class of relations for which we have decidability of the uniform strategy problem with strict quantifiers does not contain perfect recall. However our contribution shows that we can solve such games with epistemic temporal objectives if the agents have bounded memory, even if we want the knowledge of the players to be restricted to the outcomes of the distributed strategy – which forbids powerset construction techniques.

We finish this thesis with a last application of our results on uniform strategies, in the context of Dynamic Epistemic Logic.

Chapter 7

Epistemic protocol synthesis

Automated planning, as defined and studied in Ghallab et al. (2004), consists in computing a finite sequence of actions that takes some given system from its initial state to one of its designated "goal" states. The Dynamic Epistemic Logic (DEL) community has recently defined and started to investigate a special case of automated planning which is called *epistemic planning* (Bolander and Andersen, 2011; Löwe et al., 2011; Aucher and Bolander, 2013). In DEL, epistemic models and event models permit to represent very precisely how events that can occur in the world are perceived by the agents involved, and how the knowledge or beliefs of these agents concerning the world evolves with the occurrence of events. Given an initial epistemic state of the agents, a finite set of possible events and an objective property concerning the knowledge of the agents, the epistemic planning problem consists in computing a finite sequence of available events such that, after their occurrence, the objective property holds of the new state of the world. We reformulate the epistemic planning problem as a uniform strategy problem, allowing us to establish new complexity bounds for this problem, and we generalize it by defining a notion of *epistemic protocol synthesis* that we solve thanks to techniques from Chapter 5.

This chapter revolves around three frameworks: Dynamic Epistemic Logic, Epistemic Temporal Logic (ETL) and regular structures. While DEL and ETL are designed to study interactions between knowledge and time, and connections between both frameworks have already been studied (Hoshi and Yap, 2009; van Benthem et al., 2009; Aucher and Herzig, 2011; Wang and Aucher, 2013), we show that the regular structures point of view is also relevant and we establish correspondences between regular structures and the two other frameworks. Relying on the well-known embedding of DEL in ETL, we define a notion of epistemic protocol and the associated epistemic protocol synthesis problem, which generalizes in several directions the epistemic planning problem. Then we exploit our connection with regular structures to apply techniques from Chapter 5 and solve our epistemic protocol synthesis problem. We believe that our approach paves the way to apply mature and powerful automata techniques to solve problems of protocol synthesis in the framework of Dynamic Epistemic Logic.

7.1 DEL, ETL and regular structures

We start with the presentation of the three frameworks which are dynamic epistemic logic, epistemic temporal logic and regular structures. For the rest of the chapter we fix a finite nonempty set of agents Ag. We write $m = |Ag|$ the number of agents (we keep notation n available for other purposes). Like in previous chapters, AP always denotes some finite set of atomic propositions.

In this chapter we use both word automata and transducers. To avoid confusions in notations, we shall denote states of word automata as $s, s' \ldots$ and states of transducers as $q, q' \ldots$

7.1.1 Dynamic Epistemic Logic

The epistemic language $\mathcal{L}^{EL}$ is simply the language of propositional logic extended with "knowledge" modalities, one for each agent. Intuitively, $K_i\varphi$ reads as "agent i knows φ". The syntax of $\mathcal{L}^{EL}$ is given by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid K_i\varphi \qquad \text{where } p \in AP \text{ and } i \in Ag.$$

Representing the world

The semantics of $\mathcal{L}^{EL}$ is given in terms of epistemic models. Intuitively, a (pointed) epistemic model $(\mathcal{M}, w)$ represents how the agents perceive the actual world w.

Definition 45. An *epistemic model* is a tuple $\mathcal{M} = (W, \{R_i\}_{i \in Ag}, V)$ where:

- W is a non-empty finite set of possible *worlds*,
- $R_i \subseteq W \times W$ is an *accessibility relation* on W for agent $i \in Ag$,
- $V : AP \to 2^W$ is a *valuation function*.

We write $w \in \mathcal{M}$ for $w \in W$, and $(\mathcal{M}, w)$ is called a *pointed epistemic model*. In the possible worlds semantics as formalized by Hintikka (1962), $w\, R_i\, w'$ means that in world w agent i considers that w' might be the actual world. An agent then knows something if this thing is true in all the worlds that the agent considers as possibly the actual world.

Formally, given a pointed epistemic model $(\mathcal{M}, w)$, we define the semantics of $\mathcal{L}^{EL}$ by induction on its formulas:

- $\mathcal{M}, w \models p$ if $w \in V(p)$
- $\mathcal{M}, w \models \neg\varphi$ if it is not the case that $\mathcal{M}, w \models \varphi$
- $\mathcal{M}, w \models \varphi \vee \psi$ if $\mathcal{M}, w \models \varphi$ or $\mathcal{M}, w \models \psi$
- $\mathcal{M}, w \models K_i\varphi$ if for all w' such that $w\, R_i\, w'$, $\mathcal{M}, w' \models \varphi$.

To illustrate this, consider a very simple situation with two agents, Alice and Bob, where Alice places a coin in a cup, shakes the cup and puts it upside down on a table. Assume that Alice and Bob are interested in knowing whether the upside of the coin is heads or tails. In the initial situation we described, neither Alice nor Bob knows it. Figure 7.1 represents this initial epistemic situation, with atomic propositions h and t, meaning respectively heads and tails. The doubly circled world, w, is the actual world: we assume that the coin is actually on heads. The arrows represent accessibility relations

for Alice and Bob: both of them consider that the two possible worlds may be the actual one.

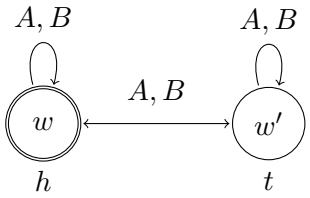

Figure 7.1: The epistemic model $\mathcal{M}$.

Letting $\mathcal{M}$ be the epistemic model in Figure 7.1, it holds that $\mathcal{M}, w \models h \wedge \neg K_A h \wedge \neg K_B h$, meaning that in world w, the coin is on heads but neither Alice nor Bob knows it.

Representing events

One of the great assets of DEL is the ability to describe in detail *events* and their occurrence in an epistemic situation. First, there may be conditions that the world must verify for an event to happen. This is represented by a function assigning to each event a *precondition*, represented as an epistemic formula. Because these preconditions are assumed to be common knowledge among the agents, the occurrence of an event provides the information that the precondition of this event was true in the world where it occurred. Second, agents may have different perceptions of the occurrence of an event. This is represented, like for epistemic models, by an accessibility relation for each agent. But an event can also modify the world in which it happens, by changing the valuations of the atomic propositions according to some *postconditions*. It is often assumed in DEL that events can only provide information and do not change the facts of the world. Such events are called *epistemic events* (Baltag et al., 1998). On the other hand, events that do modify the worlds in which they occur are called *ontic* events (see for example van Ditmarsch and Kooi (2006)). Here we consider ontic events as formalized in van Eijck (2004).

Definition 46. An *event model* (with ontic events) is a tuple $\mathcal{E} = (\mathsf{E}, \{\mathsf{R}_i\}_{i \in Ag}, \mathrm{pre}, \mathrm{post})$ where:

- E is a non-empty finite set of possible *events*,
- $\mathsf{R}_i \subseteq \mathsf{E} \times \mathsf{E}$ is an *accessibility relation* on E for agent i,
- $\mathrm{pre} : \mathsf{E} \to \mathcal{L}^{EL}$ is a *precondition function* and
- $\mathrm{post} : \mathsf{E} \to AP \to \mathcal{L}^{EL}$ is a *postcondition function*.

We write $e \in \mathcal{E}$ for $e \in \mathsf{E}$, and $(\mathcal{E}, e)$ is called a *pointed event model*, where e represents the actual event of $(\mathcal{E}, e)$. Intuitively, $e \,\mathsf{R}_i\, e'$ means that while event e is occurring, agent i considers possible that event e' is actually occurring. An event e can occur in a world w of an epistemic model $\mathcal{M}$ if, and only if, its precondition is verified, *i.e.* $\mathcal{M}, w \models \mathrm{pre}(e)$. If an event e occurs in a pointed epistemic model $(\mathcal{M}, w)$, an atomic proposition p will be true in the "updated" situation if, and only if, its postcondition is verified in the world before the event occurs, *i.e.* $\mathcal{M}, w \models \mathrm{post}(e)(p)$.

To continue our example, imagine that Alice looks under the cup. Bob sees her doing so, but does not manage to see if it is heads or tails. This can be represented with the event model in Figure 7.2. Event e_1 is Alice seeing heads, so $\text{pre}(e_1) = h$, and e'_1 is Alice seeing tails, so $\text{pre}(e'_1) = t$. Alice knows what she observes, so that she does not confuse the two events, while Bob does not know which of the two events occurs. None of these two events is ontic, so that for each event, the postcondition is the identity function id: $\text{post}(e_1)(h) = h$, $\text{post}(e_1)(t) = t$, and similarly for e'_1.

An example of ontic event would be for someone to flip the coin. Figure 7.3 represents an event model where Alice flips the coin without watching it, and Bob sees her manipulating the cup but does not know whether she flipped the coin of not. Notice that models $\mathcal{E}_1$ and $\mathcal{E}_2$ are similar in structure, however the preconditions and postconditions are different. In $\mathcal{E}_2$, e_2 corresponds to Alice flipping the coin. This can happen no matter what side of the coin is facing up, so $\text{pre}(e_2) = \texttt{true}$. However, if it was heads before the event it will be tail afterwards, and vice versa: $\text{post}(e_2)(t) = h$ and $\text{post}(e_2)(h) = t$. Event e'_2 corresponds to the case where Alice touched the coin but did not flip it: again the precondition is $\text{pre}(e'_2) = \texttt{true}$, and the postcondition is the identity function as nothing changes. Alice knows which one of the two events occurs, but Bob considers both possible.

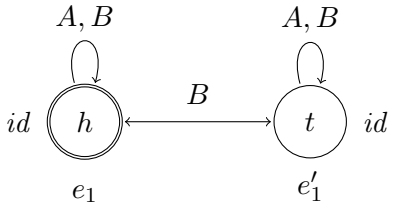

Figure 7.2: The event model $\mathcal{E}_1$.

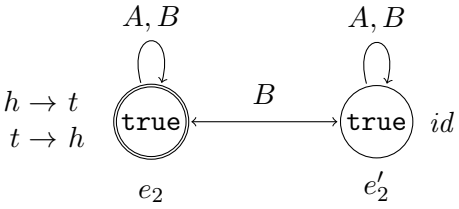

Figure 7.3: The event model $\mathcal{E}_2$.

Figure 7.4: Two event models. Inside each event is its precondition, and on the side is its postcondition.

In general, preconditions and postconditions can be epistemic formulas; however many interesting and involved situations can be modeled with only propositional pre and postconditions. In our example, notice that the preconditions and postconditions are either `true` or a unique atomic proposition. In the rest of the chapter, we say that an event model is *propositional* if both its preconditions and postconditions are propositional.

Remark 13. For technical reasons, we will assume in the following that an event model has at least one executable event for every pointed epistemic model. Formally, we assume that the disjunction of the preconditions of its events is a valid formula. When it is not the case, we may add in the model an event with trivial precondition and accessible from no other event in the model. See Remark 14 for more details on the reason of this assumption.

Updating the world

An ontic event has two effects of the world: first, it can modify the physical world itself, which is formalized by a change of the propositional valuations; second, an event can carry information impacting the agents' perception of the world, which is represented

by updating the accessibility relations of the agents. Informally, the update product of an epistemic model with an event model is a new epistemic model representing the updated world. A possible world in this model is a pair (w, e) where w is a possible world from the initial epistemic model, and e is a possible event that can occur in w. The propositional valuation of a world (w, e) is defined according to the postcondition $\text{post}(e)$, and in a world (w, e), an agent considers the world (w', e') possible if in world w she considered w' possible, and while event e has occurred she considers possible that event e' actually occurred.

Definition 47. Given an epistemic model $\mathcal{M} = (W, \{R_i\}_{i \in Ag}, V)$ and an event model $\mathcal{E} = (\mathsf{E}, \{\mathsf{R}_i\}_{i \in Ag}, \text{pre}, \text{post})$, the *update product* of $\mathcal{M}$ and $\mathcal{E}$ is the epistemic model $\mathcal{M} \otimes \mathcal{E} = (W^\otimes, \{R_i^\otimes\}_{i \in Ag}, V^\otimes)$ where:

$$\begin{aligned} W^\otimes &= \{(w, e) \in W \times \mathsf{E} \mid \mathcal{M}, w \models \text{pre}(e)\}, \\ R_i^\otimes(w, e) &= \{(w', e') \in W^\otimes \mid w' \in R_i(w) \text{ and } e' \in \mathsf{R}_i(e)\}, \\ V^\otimes(p) &= \{(w, e) \in W^\otimes \mid \mathcal{M}, w \models \text{post}(e)(p)\} \end{aligned}$$

The product of a pointed epistemic model $(\mathcal{M}, w)$ with a pointed event model $(\mathcal{E}, e)$ is $(\mathcal{M}, w) \otimes (\mathcal{E}, e) = (\mathcal{M} \otimes \mathcal{E}, (w, e))$ if $\mathcal{M}, w \models \text{pre}(e)$, and it is undefined otherwise.

Figure 7.5 illustrates the product of $\mathcal{M}$ with $\mathcal{E}_1$ from our example. According to preconditions, event e can only occur in world w, and event e' in w'. Because the postconditions are the identity function, the valuations do not change before and after the event. Finally, because Alice can tell the difference between e and e', she does not confuse (w, e) with (w', e'). As a result, one can see that after having observed heads, Alice indeed knows that the coin is on heads, while Bob did not learn anything.

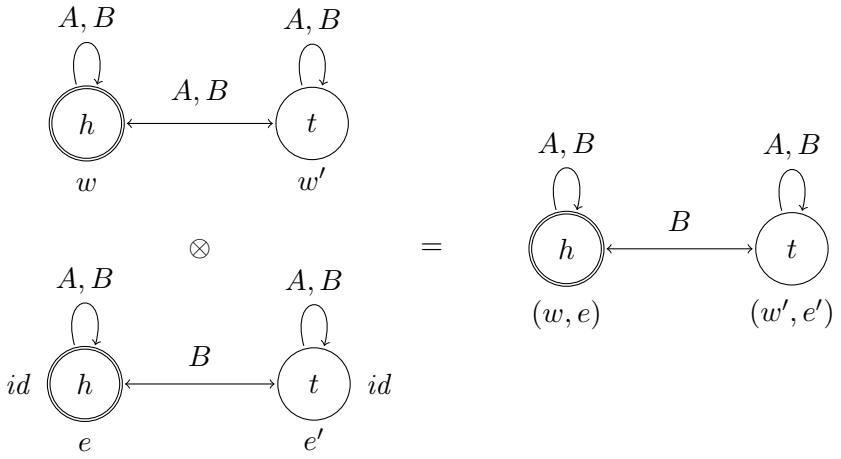

Figure 7.5: The update product of $\mathcal{M}$ and $\mathcal{E}_1$.

7.1.2 Epistemic Temporal Logic

We consider epistemic temporal models as defined in Pacuit and van Benthem (2006). Two main other approaches have been studied for modeling knowledge and time (see

Parikh and Ramanujam (1985) for the Parikh and Ramanujam framework, and Halpern and Vardi (1989); Fagin et al. (1995); Halpern et al. (2004) for interpreted systems). They are all modally equivalent (Pacuit, 2007), and we choose to use the one that resembles most the models of our language $m\mathcal{L}_{\rightsquigarrow}$.

Definition 48. An *ETL model* is a structure $\mathcal{H} = (\Sigma, H, \{\rightsquigarrow_i\}_{i\in Ag}, V)$, where Σ is a finite set of *events*, $H \subseteq \Sigma^*$ is a set of *histories* closed for non-empty prefixes, for each $i \in Ag$, $\rightsquigarrow_i$ is a binary relation on H, and $V : AP \to 2^H$ is a valuation function.

Given an ETL model $\mathcal{H} = (\Sigma, H, \{\rightsquigarrow_i\}_{i\in Ag}, V)$ we may write $h \in \mathcal{H}$ for $h \in H$.

Remark 14. We assume that every history can be continued, *i.e.* for all $h \in \mathcal{H}$, there is an event $e \in \Sigma$ such that $he \in \mathcal{H}$. This is the counterpart of the assumption in Remark 13 that for each event model, in every possible situation there is an event that can occur. These assumptions are here to fit our notion of infinite trees (see Section 2.2) and deadlock-free game arenas.

We describe how $m\mathcal{L}_{\rightsquigarrow}$ formulas can be given a semantics based on ETL models (recall that $m = |Ag|$). Given such a model $\mathcal{H} = (\Sigma, H, \{\rightsquigarrow_i\}_{i\in Ag}, V)$, we see (H, V) as a (2^{AP})-labelled Σ-forest, which plays the role of universe in the semantics of $m\mathcal{L}_{\rightsquigarrow}$. We may abuse notations and use $\mathcal{H}$ to denote this forest. We evaluate an $m\mathcal{L}_{\rightsquigarrow}$ formula in a 2^{AP}-labelled Σ-tree. It should be kept in mind that the tree in which we evaluate an $m\mathcal{L}_{\rightsquigarrow}$ formula will represent some strategy or some protocol, to be defined later. Note that a node x is a history, thus the meaning of $x \rightsquigarrow_i y$ is clear for $x, y \in t$. Note also that a branch $\lambda = x_0x_1 \ldots$ is an infinite series of histories such that for all $i \in \mathbb{N}$, $x_{i+1} = x_ie_{i+1}$ for some $e_{i+1} \in \Sigma$. Now, given a $(2^{AP}, \Sigma)$-tree $t \subseteq \mathcal{H}$, a node x of t and a state formula φ of $m\mathcal{L}_{\rightsquigarrow}$, we let $t, x \models \varphi$ denote that $\{\rightsquigarrow_i\}_{i\in Ag}, \mathcal{H}, t, x \models \varphi$ as defined in Section 6.1; we define $t, \lambda \models \psi$ similarly for a branch $\lambda \in Branches(t)$ and a path formula ψ. Again, we must remark that in $\mathcal{H}$ relations $\rightsquigarrow_i$ are defined between histories, while in the semantics of $m\mathcal{L}_{\rightsquigarrow}$ relations are defined on $(2^{AP})^*$. As explained in Remark 4, using atomic propositions that uniquely identify events and modifying the valuation V accordingly allows us to equivalently define each relation $\rightsquigarrow_i$ over $(2^{AP})^*$.

For an ETL model $\mathcal{H} = (\Sigma, H, \{\rightsquigarrow_i\}_{i\in Ag}, V)$, we define the *valuation* of a history $h \in \mathcal{H}$ as $\nu(h) = \{p \mid h \in V(p)\}$. For a propositional formula α, we shall note $h \models \alpha$ for $\nu(h) \models \alpha$, and we let $[\![\alpha]\!]^{\mathcal{H}} = \{h \in \mathcal{H} \mid h \models \alpha\}$.

We now define some properties that, as described in Section 7.2, characterize ETL models that are generated by DEL models.

Definition 49. Let $\mathcal{H} = (\Sigma, H, \{\rightsquigarrow_i\}_{i\in Ag}, V)$ be an ETL model. We define the following properties:

- *Propositional Stability (PS)*: for all $e \in \Sigma$, for all $p \in AP$, there exists a propositional formula α_p^e such that $\{h \in \mathcal{H} \mid he \in V(p)\} = [\![\alpha_p^e]\!]^{\mathcal{H}}$.
- *Synchronicity (S)*: for all $h, h' \in \mathcal{H}$, if $h \rightsquigarrow_i h'$ then $|h| = |h'|$.
- *Perfect Recall (PR)*: for all $he, h'e' \in \mathcal{H}$, if $he \rightsquigarrow_i h'e'$ then $h \rightsquigarrow_i h'$.
- *No Miracles (NM)*: for all $he, h'e', ge, g'e' \in \mathcal{H}$, if $he \rightsquigarrow_i h'e'$ and $g \rightsquigarrow_i g'$, then $ge \rightsquigarrow_i g'e'$.
- *Propositional Equivalence Invariance (PEI)*: for all histories h and h' in $\mathcal{H}$ such that $\nu(h) = \nu(h')$, $he \in \mathcal{H}$ iff $h'e \in \mathcal{H}$.

An ETL model that verifies all these properties is called a *simple* ETL model.

Observe that No Miracles for ETL models is a variant of the notion defined in Definition 27 (page 64) for binary relations on words.

Intuitively, Propositional Stability reflects the fact that the truth of atomic propositions does not evolve completely arbitrarily, but that the "physical" effects of an event are determined by the current state of the world. More precisely, it says that whether some proposition p will hold after the occurrence of an event e only depends on the atomic propositions that currently hold. Propositional Equivalence Invariance means that the same events can occur in histories that verify the same atomic propositions. Concerning the notions of Perfect Recall and No Miracles (called Uniform No Miracles in van Benthem and Liu (2004)), the definitions we take here concern the *synchronous setting*. For a discussion on the matter see for example Dégremont et al. (2011).

7.1.3 Regular structures

As discussed in Section 3.5, regular structures are infinite structures that can be represented by finite machines. They have been studied in depth in, *e.g.* , Khoussainov and Nerode (1994); Blumensath and Grädel (2000, 2004); Khoussainov et al. (2007). As we prove in the next section, they provide an intermediate step between DEL-generated ETL models (yet to be defined) and our framework for uniform strategies.

A *relational structure* is a tuple $\mathcal{S} = (D, \{\rightsquigarrow_i\}_{i\in Ag}, V)$ where D is the (possibly infinite) *domain* of $\mathcal{S}$, for each $i \in Ag$, $\rightsquigarrow_i \subseteq D \times D$ is a binary relation and $V : AP \to 2^D$ is a valuation function. In general relations can be of arbitrary arity, but here we only consider binary relations. Also, V can alternatively be seen as a set of predicate interpretations for atomic propositions in AP. Observe that epistemic models and ETL models are instances of relational structures with unary and binary relations.

Definition 50. A relational structure $\mathcal{S} = (D, \{\rightsquigarrow_i\}_{i\in Ag}, V)$ is a *regular structure* over a finite alphabet Σ if its domain $D \subseteq \Sigma^*$ is a regular language over Σ, for each i, $\rightsquigarrow_i \subseteq \Sigma^* \times \Sigma^*$ is a regular relation and for each $p \in AP$, $V(p) \subseteq D$ is a regular language. Given deterministic word automata $\mathcal{A}_\mathcal{S}$ and $\mathcal{A}_p$ ($p \in AP$), as well as transducers T_i for $i \in Ag$, we say that $(\mathcal{A}_\mathcal{S}, \{T_i\}_{i\in Ag}, \{\mathcal{A}_p\}_{p\in AP})$ is a *representation* of $\mathcal{S}$ if $\mathcal{L}(\mathcal{A}_\mathcal{S}) = D$, for each $i \in Ag$, $[T_i] = \rightsquigarrow_i$ and for each $p \in AP$, $\mathcal{L}(\mathcal{A}_p) = V(p)$.

Lemma 18. *Let $\mathcal{S} = (D, \{\rightsquigarrow_i\}_{i\in Ag}, V)$ be a regular structure. There is a representation $(\mathcal{A}_\mathcal{S}, \{T_i\}_{i\in Ag}, \{\mathcal{A}_p\}_{p\in AP})$ of $\mathcal{S}$ such that $\mathcal{A}_\mathcal{S}$ and each $\mathcal{A}_p$ share the same underlying labelled directed graph. Formally, if we note $\mathcal{A}_\mathcal{S} = (\Sigma, Q, \delta, s_\iota, F)$, then for each $p \in AP$, $\mathcal{A}_p = (\Sigma, Q, \delta, s_\iota, F_p)$ for some $F_p \subseteq Q$.*

Proof. First, let us make each $\mathcal{A}_p$ a complete automaton by sending missing transitions to a sink state. $\mathcal{A}_\mathcal{S}$ does not need to be completed as each $\mathcal{A}_p$ already recognizes a sublanguage of $\mathcal{L}(\mathcal{A}_\mathcal{S})$. Then we take the synchronized product of the labelled directed graphs underlying all these automata. This gives a labelled directed graph on which one can assign accepting states for each automaton according to the corresponding underlying states. □

Remark 15. From now on we will only consider canonical representations of regular structures of the form described in Lemma 18. For a representation $(\mathcal{A}_\mathcal{S}, \{T_i\}_{i\in Ag}, \{\mathcal{A}_p\}_{p\in AP})$ of a structure $\mathcal{S}$, where $\mathcal{A}_\mathcal{S} = (\Sigma, Q, \delta, s_\iota, F)$ and for each $p \in AP$, $\mathcal{A}_p = (\Sigma, Q, \delta, s_\iota, F_p)$, it is possible to define for each state $s \in Q$ its *valuation* $\nu_\mathcal{S}(s) = \{p \mid s \in F_p\}$. Therefore, we will consider that a canonical representation for a regular structure $\mathcal{S}$ is given as a tuple $(\mathcal{A}_\mathcal{S}, \{T_i\}_{i\in Ag}, \nu_\mathcal{S})$, where $\nu_\mathcal{S}$ is a valuation over AP for the sates of $\mathcal{A}_\mathcal{S}$, as described above.

Definition 51. Let $(\mathcal{A}_\mathcal{S}, \{T_i\}_{i\in Ag}, \nu_\mathcal{S})$ be a (canonical) representation of a regular structure $\mathcal{S}$ over Σ, and let Q be the set of states of $\mathcal{A}_\mathcal{S}$. A binary relation $R \subseteq Q \times Q$ is a *propositional bisimulation* if for all $(s, s') \in R$,

- $\nu_\mathcal{S}(s) = \nu_\mathcal{S}(s')$,
- for all $e \in \Sigma$, $\delta(s, e)$ is defined if, and only if, $\delta(s', e)$ is defined, and
- for all $e \in \Sigma$, if $\delta(s, e) = s_1$ and $\delta(s', e) = s'_1$ are defined, then $(s_1, s'_1) \in R$.

We say that two states s and s' are *propositionally bisimilar*, noted $s \leftrightarrows_{AP} s'$, if there is a propositional bisimulation R such that $(s, s') \in R$.

We define a notion of simple regular structure which, as we prove in Section 7.2, corresponds to the notion of simple ETL model. However, the properties that characterize these regular structures do not concern the domain or the relations but rather the finite state machines that recognize them.

We first define the following structural property of automata:

Definition 52. A finite state word automaton $\mathcal{A} = (\Sigma, Q, \delta, s_\iota, F)$ is said to be *nonempty-prefix-closed* if $Q = \{s_\iota\} \uplus F$, and s_ι has no incoming transition (notice that $s_\iota \notin F$).

Definition 53. Let $\mathcal{S} = (D, \{\rightsquigarrow_i\}_{i\in Ag}, V)$ be a regular structure. A representation $(\mathcal{A}_\mathcal{S}, \{T_i\}_{i\in Ag}, \nu_\mathcal{S})$ is *simple* if, letting Q be the set of states of $\mathcal{A}_\mathcal{S}$:

1. $\mathcal{A}_\mathcal{S}$ is a nonempty-prefix-closed deterministic automaton with no deadlock [1],
2. for all $i \in Ag$, there is a one-state synchronous transducer T'_i such that, letting T_D be the transducer recognizing the identity relation on D, $T_i = T_D \circ T'_i \circ T_D$,
3. for all $s, s' \in Q$, if $\nu_\mathcal{S}(s) = \nu_\mathcal{S}(s')$ then $s \leftrightarrows_{AP} s'$.

$\mathcal{S}$ is a *simple* regular structure if it has a simple representation.

Property 1 just ensures that the domain of the structure is the domain of some ETL model. Property 2 says that each relation is intrinsically recognized by a very simple transducer, and it ensures that the relation verifies Synchronicity, Perfect Recall and No Miracles. Finally, Property 3 ensures both Propositional Equivalence Invariance and Propositional Stability.

7.2 Merging frameworks

In this section we build bridges between the three frameworks of DEL, ETL and regular structures. The connections between DEL and ETL (in the synchronous setting) are well

1. Classically, we call deadlock a state with no outgoing transition.

known (van Benthem et al., 2009; Wang and Aucher, 2013), we just adapt them to the case of propositional event models with ontic events. As a new result, however, we show that relational structures generated by propositional DEL models are regular structures. Using this result, we establish a connection between "DEL-generated" ETL models and the framework of uniform strategies, which enables us to solve our problem of epistemic protocol synthesis in Section 7.3.

7.2.1 DEL and ETL

van Benthem and Liu (2004) show how an iterative application of event models to an initial epistemic model yields a structure that can be seen as an ETL model, and conversely how ETL models that verify certain properties can be seen as "DEL-generated", the meaning of which we now make precise. We focus on a rather simple case where a single event model is repeatedly applied to some initial epistemic model, but we shall see that this suffices to model interesting problems.

Definition 54. For an epistemic model $\mathcal{M} = (W, \{R_i\}_{i\in Ag}, V)$ and an event model $\mathcal{E} = (\mathsf{E}, \{\mathsf{R}_i\}_{i\in Ag}, \mathrm{pre}, \mathrm{post})$, we define the family of epistemic models $\{\mathcal{ME}^n\}_{n\geq 0}$ by letting $\mathcal{ME}^0 = \mathcal{M}$ and $\mathcal{ME}^{n+1} = \mathcal{ME}^n \otimes \mathcal{E}$. Letting, for each n, $\mathcal{ME}^n = (W^n, \{R_i^n\}_{i\in Ag}, V^n)$, we define the ETL model $\mathcal{ME}^* = (\Sigma, H, \{\rightsquigarrow_i\}_{i\in Ag}, V)$, where:

- $\Sigma = W \cup \mathsf{E}$,
- $H = \bigcup_{n\geq 0} W^n$,
- $h \rightsquigarrow_i h'$ if there is some n such that $h, h' \in \mathcal{M}^n$ and $h\, R_i^n\, h'$, and
- $V(p) = \bigcup_{n\geq 0} V^n(p)$.

A history $h \in H$ is thus an element $(w, e_1, \ldots, e_n) \in \mathcal{ME}^n$ for some n, that we simply write $we_1 \ldots e_n$. Note that the set of histories that we define is indeed closed for non-empty prefixes. Also, because by assumption every event model has at least one executable event for each possible situation (Remark 13), every history can be continued. Therefore $\mathcal{ME}^*$ is an ETL model (Remark 14).

In the rest of the chapter we shall say that an ETL model is *DEL-generated* if it is equal to $\mathcal{ME}^*$ for some epistemic model $\mathcal{M}$ and some event model $\mathcal{E}$. If in addition the event model $\mathcal{E}$ is propositional, we shall say that the ETL model is *PDEL-generated*. Note that all PDEL-generated models are DEL-generated, but the contraposite does not hold.

Remark 16. Because we will sometimes need to track worlds and events in a DEL-generated ETL model, we assume that when an epistemic model $\mathcal{M}$ is given, for each world $w \in \mathcal{M}$ there is an atomic proposition $p_w \in AP$ such that for every world $w' \in \mathcal{M}$, $\mathcal{M}, w' \models p_w$ if and only if $w = w'$. In addition we assume that for an event model $\mathcal{E}$, for each event $e \in \mathcal{E}$ there is an atomic proposition $p_e \in AP$ such that the postconditions of $\mathcal{E}$ verify:

$$\text{for all } e, e' \in \mathcal{E} \text{ and } w \in \mathcal{M},$$
$$\mathrm{post}(e)(p_w) = \texttt{false} \qquad \mathrm{post}(e')(p_e) = \begin{cases} \texttt{true} & \text{if } e = e', \\ \texttt{false} & \text{otherwise.} \end{cases}$$

As we will always consider a finite number of epistemic and event models at a time, these assumptions comply with the finiteness of AP.

Lemma 19. *Let $\mathcal{M}$ be an epistemic model, and let $\mathcal{E}$ be an event model. For all $w \in \mathcal{M}$, for all $e \in \mathcal{E}$ and history $w_0 e_1 \dots e_n \in \mathcal{M}\mathcal{E}^*$, it holds that:*

$$w_0 e_1 \dots e_n \models p_w \text{ if, and only if, } n = 0 \text{ and } w_0 = w, \text{ and}$$
$$w_0 e_1 \dots e_n \models p_e \text{ if, and only if, } n > 0 \text{ and } e_n = e.$$

Proof. The proof is a direct application of the assumption made in Remark 16 and the definition of $\mathcal{M}\mathcal{E}^*$. □

The following proposition is a slight modification of the one in van Benthem and Liu (2004). It was originally stated without ontic events, and with arbitrary preconditions instead of propositional ones; therefore they use a stronger notion of propositional stability, and epistemic bisimilarity invariance instead of propositional equivalence invariance.

Proposition 26. *Let $\mathcal{H}$ be an ETL model. There exist an epistemic model $\mathcal{M}$ and a* propositional *event model $\mathcal{E}$ such that $\mathcal{H} = \mathcal{M}\mathcal{E}^*$ if and only if $\mathcal{H}$ is a simple ETL model (*i.e. *satisfies properties PS, S, PR, NM and PEI of Definition 49).*

Proof. The proof is a simple adaptation of van Benthem and Liu (2004).

Left-to-right implication. Let $\mathcal{M} = (W, \{R_i\}_{i \in Ag}, V)$ be an epistemic model, let $\mathcal{E} = (\mathsf{E}, \{\mathsf{R}_i\}_{i \in Ag}, \mathrm{pre}, \mathrm{post})$ be a propositional event model, and let $\mathcal{H} = \mathcal{M}\mathcal{E}^*$ be the ETL model over $\Sigma = W \cup \mathsf{E}$ generated by $\mathcal{M}$ and $\mathcal{E}$. By definition of the update product $\otimes$ and by construction of $\mathcal{M}\mathcal{E}^*$, checking that $\mathcal{H}$ verifies S, PR and NM is routine. For propositional stability, take $e \in \Sigma$, $p \in AP$ and $h \in H$. By definition of the update product, $he \in V(p)$ if and only if $he \in H$ and $h \models \mathrm{post}(e)(p)$. Because preconditions and postconditions are propositional, we can let $\alpha_p^e := \mathrm{pre}(e) \wedge \mathrm{post}(e)(p)$, and we have that $\{h \in H \mid he \in V(p)\} = [\![\alpha_p^e]\!]^{\mathcal{H}}$. Propositional Equivalence Invariance comes from the fact that all preconditions of events in $\mathcal{E}$ are propositional: if two histories verify the same atomic propositions, the same events can extend them.

Right-to-left implication. let $\mathcal{H} = (\Sigma, H, \{\rightsquigarrow_i\}_{i \in Ag}, V)$ be an ETL model over the finite set of events Σ that verifies PS, S, PR, NM and PEI. We define the epistemic model $\mathcal{M} = (W, R, V_{\mathcal{M}})$ where the set of worlds $W = \Sigma \cap H$ is the set of histories of length one, each R_i is the restriction of $\rightsquigarrow_i$ to W and $V_{\mathcal{M}}$ is the restriction of V to W. The event model $\mathcal{E} = (\mathsf{E}, \mathsf{R}, \mathrm{pre}, \mathrm{post})$ is defined as follows. $\mathsf{E} = \Sigma$ is the set of possible events, and $e \,\mathsf{R}_i\, e'$ if there are he and $h'e'$ in H such that $he \rightsquigarrow_i h'e'$. Regarding preconditions, since $\mathcal{H}$ verifies Propositional Equivalence Invariance, for each event e, the set $\{h \mid he \in H\}$ is closed by propositional equivalence (within H). This set is thus definable by a simple propositional formula $\mathrm{pre}(e)$, and because each h in $\mathcal{H}$ can be extended with at least one event, these formulas can be chosen so that $\vee_{e \in \Sigma} \mathrm{pre}(e) \equiv \mathtt{true}$. The event model $\mathcal{E}$ thus verifies the assumption of Remark 13. For the postconditions, because $\mathcal{H}$ verifies propositional stability, for each $e \in \Sigma$ and $p \in AP$ there exists a propositional formula α_p^e such that $\{h \in H \mid he \in V(p)\} = [\![\alpha_p^e]\!]^{\mathcal{H}}$. We let $\mathrm{post}(e)(p) := \alpha_p^e$. With $\mathcal{M}$ and $\mathcal{E}$ thus defined, we can prove that $\mathcal{H} = \mathcal{M}\mathcal{E}^*$.

Let us note $\mathcal{M}\mathcal{E}^* = (\Sigma, H', \{\rightsquigarrow_i'\}_{i \in Ag}, V')$. First, it is not hard to see that $H = H'$(for example by induction on the length of histories). Now, let $p \in AP$. We prove that $h \in V(p)$ iff $h \in V'(p)$. For $|h| = 1$, $h \in \mathcal{M}$ by definition of $\mathcal{M}\mathcal{E}^*$, and the result follows from the definition of $\mathcal{M}$. Now if $h = h'e \in H$, $h'e \in V(p)$ iff $h' \models \alpha_p^e$ iff $h' \models \mathrm{post}(e)(p)$ iff $h'e \in V'(p)$.

Finally, we prove by induction on the common length n of h and h' that $h \rightsquigarrow_i h'$ iff $h \rightsquigarrow'_i h'$. For $n = 1$ the result follows directly from the definitions of $\mathcal{M}$ and $\mathcal{M}\mathcal{E}^*$. For $n+1$, suppose that $he \rightsquigarrow_i h'e'$. By Perfect Recall, we have that $h \rightsquigarrow_i h'$, and by induction hypothesis, $h \rightsquigarrow'_i h'$. Because he and $h'e'$ are in the domain, we have that $h \models \mathrm{pre}(e)$ and $h' \models \mathrm{pre}(e')$. And because $he \rightsquigarrow_i h'e'$, by definition of $\mathcal{E}$ we have that $e \,\mathsf{R}_i\, e'$, hence $he \rightsquigarrow'_i h'e'$. Now assume that $he \rightsquigarrow'_i h'e'$. By definition of the update product we have that, (i) $h \rightsquigarrow'_i h'$, and (ii) $e \,\mathsf{R}_i\, e'$ in $\mathcal{E}$. Point (i) implies by induction hypothesis that $h \rightsquigarrow_i h'$, and Point (ii) implies by definition of R_i that there are ge and $g'e'$ in H such that $ge \rightsquigarrow_i g'e'$. This, together with No Miracles and the fact that $h \rightsquigarrow_i h'$, makes that $he \rightsquigarrow_i h'e'$. □

We also establish that the semantics of $\mathcal{L}^{EL}$ is preserved by the translation from DEL to ETL. To this aim we first define the following mapping from $\mathcal{L}^{EL}$ to $\mathcal{F}m\mathcal{L}_{\rightsquigarrow}$:

Definition 55. Given a $\mathcal{L}^{EL}$-formula φ, we inductively define the $\mathcal{F}m\mathcal{L}_{\rightsquigarrow}$ formula $\widetilde{\varphi}$ as follows:

$$\widetilde{p} = p \quad \widetilde{\neg\varphi} = \neg\widetilde{\varphi} \quad \widetilde{\varphi_1 \vee \varphi_2} = \widetilde{\varphi_1} \vee \widetilde{\varphi_2} \quad \widetilde{K_i\varphi} = \boxdot_i\widetilde{\varphi}.$$

We may abuse vocabulary by talking about the $\boxdot$-depth $d(\varphi)$ of an $\mathcal{L}^{EL}$ formula φ, by letting $d(\varphi) = d(\widetilde{\varphi})$, and similarly for alternation depth, $ad(\varphi)$.

Notice that we translate knowledge operators into *full* quantifiers. Because in the semantics of DEL it is implicitely assumed that agents do not know more of what happens than what is represented in the event models, and in particular they do not know what the protocol that generates a given sequence of events is, the full semantics is the one that correctly captures the agents' knowledge. We formalize this idea with the following lemma.

Lemma 20. *Let $\mathcal{M}$ and $\mathcal{E}$ be respectively an epistemic model and an event model, and let $n \in \mathbb{N}$. For every world $(w, e_1, \ldots, e_n) \in \mathcal{M} \otimes \mathcal{E}^n$, every epistemic formula $\varphi \in \mathcal{L}^{EL}$ and every tree $t \subseteq \mathcal{M}\mathcal{E}^*$ such that $we_1 \ldots e_n \in t$,*

$$\mathcal{M} \otimes \mathcal{E}^n, (w, e_1, \ldots, e_n) \models \varphi \text{ iff } \mathcal{M}\mathcal{E}^*, t, we_1 \ldots e_n \models \widetilde{\varphi}.$$

Proof. This follows directly from the definition of $\mathcal{M}\mathcal{E}^*$ (Definition 54) and the semantics of $m\mathcal{L}_{\rightsquigarrow}$ (note that there are no temporal operators in the formulas φ and $\widetilde{\varphi}$). □

Remark 17. In Section 7.3.2 we also consider agents who know the protocol, in which case their knowledge is captured using the strict semantics. Roughly, for agents who ignore the protocol, our results on fully-uniform strategies allow us to prove the decidability of the epistemic protocol synthesis problem – that we define in Section 7.3.2 – with the classic perfect-recall assumption, inherent to DEL. However, for agents who know the protocol, our results on strictly-uniform strategies only allow us to prove decidability of the epistemic protocol synthesis problem if agents have bounded memory.

This achieves the connection between ETL and DEL.

7.2.2 PDEL-generated models and regular structures

We now establish correspondences between PDEL-generated ETL models and regular structures.

Theorem 20. *Let $\mathcal{H}$ be an ETL model. $\mathcal{H}$ is a simple regular structure if, and only if, it is PDEL-generated.*

The rest of the section is dedicated to the proof of Theorem 20. The left-to-right implication results from the following lemma together with Proposition 26.

Lemma 21. *If $\mathcal{S}$ is a simple regular structure, then it is a simple ETL model.*

Proof. Let $\mathcal{S} = (D, \{\rightsquigarrow_i\}_{i\in Ag}, V)$ be a simple regular structure, and take a simple representation $(\mathcal{A}_\mathcal{S}, \{T_i\}_{i\in Ag}, \nu_\mathcal{S})$ (see Definition 53). Because $\mathcal{A}_\mathcal{S}$ is nonempty-prefix-closed (Property 1 of Definition 53), it is not hard to see that D is closed for nonempty prefixes, and because $\mathcal{A}_\mathcal{S}$ has no deadlock (Property 1 of Definition 53) every element in D can be extended in D; by Definition 48, $\mathcal{S}$ is an ETL model. Because for each $i \in Ag$, $\rightsquigarrow_i$ is the restriction to D of the relation recognized by a one-state synchronous transducer (Property 2 of Definition 53), $\mathcal{S}$ verifies Synchronicity, Perfect Recall and No Miracles. It remains to exploit Property 3 of Definition 53 and prove that $\mathcal{S}$ verifies Propositional Stability and Propositional Equivalence Invariance.

Let $\mathcal{A}_\mathcal{S} = (\Sigma, Q, \delta, s_\iota, F)$. First, observe that for each $h \in D$, $s = \delta(s_\iota, h)$ is defined, $s \neq s_\iota$ and $\nu(h) = \nu_\mathcal{S}(s)$.

Now, for Propositional Stability, we define for each atomic proposition $p \in AP$ and event $e \in \Sigma$ the set $\mathrm{Pred}(p, e) = \{s \mid p \in \nu_\mathcal{S}(\delta(s, e))\}$. $\mathrm{Pred}(p, e)$ is the set of states in $\mathcal{A}_\mathcal{S}$ from which reading e takes $\mathcal{A}_\mathcal{S}$ to a state whose valuation contains p. We clearly have:

$$\{h \in D \mid he \in V(p)\} = \{h \in D \mid \delta(s_\iota, h) \in \mathrm{Pred}(p, e)\}.$$

One can build from the set of valuations $\{\nu_\mathcal{S}(s) \mid s \in \mathrm{Pred}(p, e)\}$ a propositional formula α^e_p that is verified exactly by this set of valuations. Formally, we let:

$$\alpha^e_p = \bigvee_{s\in\mathrm{Pred}(p,e)} \bigwedge_{p'\in\nu_\mathcal{S}(s)} p' \wedge \bigwedge_{p'\notin\nu_\mathcal{S}(s)} \neg p'.$$

We prove that $\{h \in D \mid he \in V(p)\} = [\![\alpha^e_p]\!]^\mathcal{S}$, or equivalently, that:

$$\{h \in D \mid \delta(s_\iota, h) \in \mathrm{Pred}(p, e)\} = [\![\alpha^e_p]\!]^\mathcal{S}.$$

For the left-to-right inclusion, take $h \in D$ such that $\delta(s_\iota, h) \in \mathrm{Pred}(p, e)$, and note $s = \delta(s_\iota, h)$. By definition of α^e_p, because $s \in \mathrm{Pred}(p, e)$ it holds that $\nu_\mathcal{S}(s) \models \alpha^e_p$, and because $\nu(h) = \nu_\mathcal{S}(s)$, we have $h \in [\![\alpha^e_p]\!]^\mathcal{S}$.

Now for the right-to-left inclusion, take $h \in [\![\alpha^e_p]\!]^\mathcal{S}$. Writing $s = \delta(s_\iota, h)$, we have that $\nu(h) = \nu_\mathcal{S}(s)$. By definition of α^e_p, we also have that there exists $s' \in \mathrm{Pred}(p, e)$ such that $\nu(h) = \nu_\mathcal{S}(s')$. Because $\nu_\mathcal{S}(s) = \nu_\mathcal{S}(s')$, we have by Property 3 of Definition 53 that $s \leftrightarrows_{AP} s'$. From this and the fact that $s' \in \mathrm{Pred}(p, e)$, we conclude that $s \in \mathrm{Pred}(p, e)$.

For Propositional Equivalence Invariance, take $h, h' \in D$ such that $\nu(h) = \nu(h')$. Letting $s = \delta(s_\iota, h)$ and $s' = \delta(s_\iota, h')$ we have that $\nu_\mathcal{S}(s) = \nu(h) = \nu(h') = \nu_\mathcal{S}(s')$, and

again because $\mathcal{S}$ verifies Property 3 of Definition 53, $s \leftrightarrows_{AP} s'$. This implies that for all events e, $\delta(s,e)$ is defined if, and only if, $\delta(s',e)$ is defined. Because $\mathcal{A}_{\mathcal{S}}$ has only accepting states (except for the initial state s_ι), it follows that if $\delta(s,e)$ and $\delta(s',e)$ are defined, they both belong to F. Finally, $he \in D$ if, and only if, $h'e \in D$. □

The second lemma below proves the right-to-left implication of Theorem 20.

Lemma 22. *If $\mathcal{M}$ is an epistemic model and $\mathcal{E}$ is a propositional event model, $\mathcal{H} = \mathcal{M}\mathcal{E}^*$ is a simple regular structure. It has a simple representation $(\mathcal{A}_{\mathcal{H}}, \{T_i\}_{i\in Ag}, \nu_{\mathcal{H}})$ such that $|\mathcal{A}_{\mathcal{H}}| \leq 2^{|AP|} \cdot (|\mathcal{M}| + |\mathcal{E}|)$.*

Proof. Let $\mathcal{M} = (W, R, V)$ be an epistemic model, let $\mathcal{E} = (\mathsf{E}, \mathsf{R}, \mathrm{pre}, \mathrm{post})$ be a propositional event model, and let $\mathcal{H} = \mathcal{M}\mathcal{E}^* = (\Sigma, H, \{\rightsquigarrow_i\}_{i\in Ag}, V_H)$.

Define the word automaton $\mathcal{A}_{\mathcal{H}} = (\Sigma, Q, \delta, s_\iota, F)$, where $\Sigma = W \cup \mathsf{E}$, $F = \{s_\nu \mid \nu \subseteq AP\}$ and $Q = F \uplus \{s_\iota\}$. For a world $w \in W$, we define its valuation as $\nu(w) := \{p \in AP \mid w \in V(p)\}$. We now define δ, which is the following partial transition function:

$$\begin{array}{ll} \forall w \in W,\ \forall e \in \mathsf{E}, & \\ \delta(s_\iota, w) = s_{\nu(w)} & \delta(s_\iota, e) \text{ is undefined,} \\ \delta(s_\nu, w) \text{ is undefined} & \delta(s_\nu, e) = \begin{cases} s_{\nu'}, \text{ with } \nu' = \{p \mid \nu \models \mathrm{post}(e)(p)\} & \text{if } \nu \models \mathrm{pre}(e) \\ \text{undefined} & \text{otherwise.} \end{cases} \end{array}$$

It is not hard to see that $\mathcal{L}(\mathcal{A}_{\mathcal{H}}) = H$, hence H is a regular language. Also $\mathcal{A}_{\mathcal{H}}$ is nonempty-prefix closed, and because of Remark 13 it is deadlock-free.

Concerning valuations, take some $p \in AP$. Let $\mathcal{A}_p = (\Sigma, Q, \delta, s_\iota, F_p)$, where $F_p = \{s_\nu \mid p \in \nu\}$. Clearly, $\mathcal{L}(\mathcal{A}_p) = V_H(p)$, hence $V_H(p)$ is a regular language. For a state $s \in Q$, define $\nu_{\mathcal{H}}(s) = \{p \mid s \in F_p\}$.

Observe that $\mathcal{A}_{\mathcal{H}}$ has $2^{|AP|}$ states and less than $|W|+|\mathsf{E}|$ transitions per state. Therefore $|\mathcal{A}_{\mathcal{H}}| \leq 2^{|AP|} \cdot (|\mathcal{M}| + |\mathcal{E}|)$.

For the relations, let $i \in Ag$ and consider the one-state synchronous transducer $T_i = (\Sigma, Q' = \{q\}, \Delta_i, s, F' = \{q\})$, where $\Delta_i = \{(q, w, w', q) \mid w\, R_i\, w'\} \cup \{(q, e, e', q) \mid e\, \mathsf{R}_i\, e'\}$. It is easy to see that $\rightsquigarrow_i = [T_i] \cap H \times H$. Since $[T_i]$ is a regular relation and H is a regular language, $\rightsquigarrow_i$ is a regular relation recognized by $T_i' = T_{\mathcal{A}_{\mathcal{H}}} \circ T_i \circ T_{\mathcal{A}_{\mathcal{H}}}$. Therefore, $\mathcal{M}\mathcal{E}^*$ is a regular structure, and $(\mathcal{A}_{\mathcal{H}}, \{T_i'\}_{i\in Ag}, \nu_{\mathcal{H}})$ is a representation that verifies Property 1 and Property 2 of Definition 53. It remains to observe that by construction, there are not two states in $\mathcal{A}_{\mathcal{H}}$ that have the same valuation, hence $(\mathcal{A}_{\mathcal{H}}, \{T_i'\}_{i\in Ag}, \nu_{\mathcal{H}})$ also verifies Property 3, and this concludes the proof. □

We obtain as a direct corollary of Theorem 20:

Corollary 13. *An ETL model is simple if, and only if, it is a simple regular structure.*

7.2.3 PDEL-generated ETL models and game arenas

The correspondence of the last section between simple ETL models – or equivalently PDEL-generated ETL models, by Proposition 20 – and simple regular structures, though being interesting in itself, is just a step in our process that aims at computing epistemic protocols (still to be defined) thanks to uniform strategies techniques. We now use this

connection to define an ultimate translation, from PDEL-generated models to the framework of uniform strategies.

Proposition 27. *Let $\mathcal{H} = \mathcal{M}\mathcal{E}^*$ be a PDEL-generated ETL model, and let $w_\iota \in \mathcal{M}$ be an initial possible world. There are a labelled game arena $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$ and transducers $\{T_i\}_{i\in Ag}$ over 2^{AP} such that:*

1. *$|\mathcal{G}| = 2^{|AP|} \cdot (|\mathcal{M}| + |\mathcal{E}|)$,*
2. *every T_i has one state,*
3. *there is a bijection $f : \mathcal{H} \to Paths_*(V_\iota)$,*
4. *the restriction of f to $\mathcal{H}_{w_\iota}$ [2] is a bijection between $\mathcal{H}_{w_\iota}$ and $Plays_*$, and*
5. *for every $(2^{AP}, \Sigma)$-tree $t \subseteq \mathcal{H}$, for every node $x \in t$, for every state formula $\varphi \in m\mathcal{L}_{\rightsquigarrow}$, $t, x \models \varphi$ iff $f(t), f(x) \models \varphi$.*
6. *Furthermore, if accessibility relations in $\mathcal{M}$ and $\mathcal{E}$ are reflexive, then for each $i \in Ag$, $[T_i]$ verifies No Miracles.*

Proof. The idea is to build a game arena with only one player, Player 1, who just chooses events one after another. The arena, as well as the transducers that recognize the relations $\rightsquigarrow_i$, are obtained almost directly by our correspondence between PDEL-generated ETL models and simple regular structures.

Let $\mathcal{M} = (W, \{R_i\}_{i\in Ag}, V)$ be an epistemic model, let $\mathcal{E} = (\mathsf{E}, \{\mathsf{R}_i\}_{i\in Ag}, \mathrm{pre}, \mathrm{post}))$ be a propositional event model and let $\mathcal{H} = \mathcal{M}\mathcal{E}^* = (\Sigma, H, \{\rightsquigarrow_i\}_{i\in Ag}, V_\mathcal{H})$, where $\Sigma = W \cup \mathsf{E}$. Also, let $w_\iota \in \mathcal{M}$ be an initial possible world. By Theorem 20, because $\mathcal{H}$ is PDEL-generated it is a simple regular structure, and more precisely, by Lemma 22 it has a simple representation $(\mathcal{A}_\mathcal{H}, \{T_i\}_{i\in Ag}, \nu_\mathcal{H})$ such that $|\mathcal{A}_\mathcal{H}| \leq 2^{|AP|} \cdot (|\mathcal{M}| + |\mathcal{E}|)$ and for each $i \in Ag$, T_i has one state. Let $\mathcal{A}_\mathcal{H} = (\Sigma, Q, \delta, s_\iota, F)$.

The game arena that we aim at defining is essentially the automaton $\mathcal{A}_\mathcal{H}$. However, in the automaton $\mathcal{A}_\mathcal{H}$ the events are on the transitions, while in our formalism game arenas have unlabelled moves. We therefore take *transitions* of $\mathcal{A}_\mathcal{H}$ as positions in our game arena. Formally, we define the one-player labelled game arena $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$, where:

- $V = \{(s, e, s') \mid s, s' \in Q, e \in \Sigma \text{ and } s' = \delta(s, e) \text{ is defined}\}$
- $E = \{((s, e, s'), (s'', e', s''')) \mid s' = s''\}$
- $V_\iota = \{(s_\iota, w, \delta(s_\iota, w)) \mid w \in W\}$
- $v_\iota = (s_\iota, w_\iota, \delta(s_\iota, w_\iota))$
- $\mu(s, e, s') = \nu_\mathcal{H}(s')$.

Notice that $\mathcal{G}$ has the same size as $\mathcal{A}_\mathcal{H}$. In $\mathcal{G}$, a play necessarily starts with the initial possible world w_ι, so the initial position is the transition of $\mathcal{A}_\mathcal{H}$ reading w_ι in its initial state s_ι. However, because agents may not know the initial world w_ι, or in other words $\rightsquigarrow_i$ related histories do not necessarily start with w_ι, we allow the full quantifiers $\boxdot_i$ to range over all paths in the arena that start with some world $w \in W$. This is captured by letting the set of starting positions V_ι be the set of transitions of $\mathcal{A}_\mathcal{H}$ reading some possible world of $\mathcal{M}$ in its initial state.

Because $\mathcal{A}_\mathcal{H}$ is deterministic and $H = \mathcal{L}(\mathcal{A}_\mathcal{H})$, there is a bijection f_1 between H and accepting runs of $\mathcal{A}_\mathcal{H}$. Now because $\mathcal{A}_\mathcal{H}$ is a nonempty-prefix closed automaton (see

2. The set of histories that start with w_ι (see Definition 3, page 13).

Definition 52), all runs (on nonempty words) are accepting. Therefore f_1 is a bijection between H and nonempty runs of $\mathcal{A}_{\mathcal{H}}$. Now, by construction of $\mathcal{G}$, there is clearly a bijection f_2 between nonempty runs of $\mathcal{A}_{\mathcal{H}}$ and $Paths_*(V_\iota)$ – recall that because every history $h \in H$ starts with a world $w \in W$, the only symbols that $\mathcal{A}_{\mathcal{H}}$ can read in its initial state are $w \in W$. Defining $f = f_2 \circ f_1$ gives the bijection required by Property 3 of Proposition 27. Property 4 follows directly from the definition of the initial position v_ι.

The following lemma shows that the labelling of positions in the arena is correct:

Fact 5. *For a history $h \in \mathcal{H}$, letting $v = last(f(h)) \in V$, we have that $\nu(h) = \mu(v)$.*

Proof. Let $h = we_1 \dots e_n \in \mathcal{H}$ and note $f(h) = (s_\iota, w, s_0)(s_0, e_1, s_1) \dots (s_{n-1}, e_n, s_n)$. By definition of the arena, we have that $s_n = \delta(s_\iota, h)$, and because $(\mathcal{A}_{\mathcal{H}}, \{T_i\}_{i \in Ag}, \nu_{\mathcal{H}})$ is a canonical representation of $\mathcal{H}$ (see Remark 15), we have that $\nu(h) = \nu_{\mathcal{H}}(\delta(s_\iota, h)) = \nu_{\mathcal{H}}(s_n)$. By definition of the arena, $\mu(s_{n-1}, e_n, s_n) = \nu_{\mathcal{H}}(s_n)$, therefore $\mu(s_{n-1}, e_n, s_n) = \nu(h)$, which concludes. □

We can now prove Property 5 of Proposition 27. By Property 2 of Definition 53, for each $i \in Ag$ there is a one-state synchronous transducer T'_i such that $T_i = T_H \circ T'_i \circ T_H$, where T_H recognizes the identity relation on H. We adapt each T'_i so that it works on alphabet 2^{AP}.

First, remember that every world $w \in \mathcal{M}$ (resp. event $e \in \mathcal{E}$) is identified by a dedicated atomic proposition p_w (resp. p_e) (see Remark 16).

For each $i \in Ag$, letting $T'_i = (\Sigma, \{q^i\}, \Delta^i, \{q^i\}, \{q^i\})$, we can define the transducer $\widetilde{T'_i} = (2^{AP}, \{q^i\}, \widetilde{\Delta^i}, \{q^i\}, \{q^i\})$, where for all $\nu, \nu' \in 2^{AP}$:

$$(q^i, \nu, \nu', q^i) \in \widetilde{\Delta^i} \quad \text{if} \quad \begin{cases} p_w \in \nu,\ p_{w'} \in \nu' \text{ and } (q^i, w, w', q^i) \in \Delta^i, & \text{or} \\ p_e \in \nu,\ p_{e'} \in \nu' \text{ and } (q^i, e, e', q^i) \in \Delta^i, \end{cases}$$

With $\widetilde{T'_i}$ thus defined, one can prove thanks to Lemma 19 and Fact 5 that for two histories $h, h' \in \mathcal{H}$, letting $\rho = f(h)$ and $\rho' = f(h')$, $h \sim_i h'$ if and only if $\mu(\rho)[\widetilde{T'_i}]\mu(\rho')$ [3]. From this, Property 5 of Proposition 27 is easily proved by induction on the formula, and all cases are routine.

Finally, if the accessibility relations in $\mathcal{M}$ and $\mathcal{E}$ are reflexive, then for all $i \in Ag$ we have that for all $w \in \mathcal{M}$ and $e \in \mathcal{E}$, $(q^i, w, w, q^i) \in \Delta^i$ and $(q^i, e, e, q^i) \in \Delta^i$. Therefore, $[T'_i]$ satisfies No Miracles, and thus so does $[\widetilde{T'_i}]$. This proves Property 6 of Proposition 27. □

7.3 Epistemic protocol synthesis

We first recall the definition of the epistemic planning problem as well as the main results about it. Then, exploiting the connection established in Section 7.2.2 between PDEL-generated ETL models and regular structures, we manage to resort to our techniques developed for the fully-uniform strategy problem in order to solve the epistemic planning problem in the case of propositional events. This gives an alternative decidability proof for this problem, it provides upper bounds on its time complexity, and our

3. recall that for a partial play ρ, $\mu(\rho)$ is the sequence of labellings of its positions.

decision procedure can synthesize as a by-product a word automaton that represents the set of all solution plans.

In a second time, relying on the embedding of DEL into ETL, we define a problem that subsumes the epistemic planning problem and that we call the *epistemic protocol synthesis problem*. The same approach that we use for solving the epistemic planning problem allows us to solve our epistemic protocol synthesis problem. Again, the fact that our decision procedures for the existence of uniform strategies enable the synthesis of a uniform strategy when there exists one – see Pages 59 and 79 – , our procedures to decide the existence of epistemic protocols also synthesize a solution protocol whenever one exists.

7.3.1 Epistemic planning

In the community of Dynamic Epistemic Logic, the problem of epistemic planning (Bolander and Andersen, 2011; Löwe et al., 2011) is usually defined as follows.

Definition 56 (Epistemic planning problem)**.** Given an initial pointed epistemic model $(\mathcal{M}_\iota, w_\iota)$, a finite set $\mathfrak{E}$ of pointed event models and an epistemic goal formula $\Phi \in \mathcal{L}^{EL}$, decide if there is a finite series of pointed event models $(\mathcal{E}_1, e_1), \ldots, (\mathcal{E}_n, e_n) \in \mathfrak{E}$ such that $(\mathcal{M}_\iota, w_\iota) \otimes (\mathcal{E}_1, e_1) \otimes \ldots \otimes (\mathcal{E}_n, e_n) \models \Phi$.

For an instance $((\mathcal{M}_\iota, w_\iota), \mathfrak{E}, \Phi)$ of the epistemic planning problem we define its *size* as the sum of the sizes of its components, plus the number of atomic propositions used: $|((\mathcal{M}_\iota, w_\iota), \mathfrak{E}, \Phi)| = |\mathcal{M}_\iota| + \sum_{(\mathcal{E},e)\in\mathfrak{E}} |\mathcal{E}| + |\Phi| + |AP|$, where the size of an epistemic or event model is the number of edges in its graph.

The epistemic planning problem is undecidable (Bolander and Andersen, 2011; Aucher and Bolander, 2013). However, Bolander and Andersen (2011) proved that the problem is decidable in the case of one agent and equivalence accessibility relations in epistemic and event models. More recently, Aucher and Bolander (2013) and Yu et al. (2013) proved independently that the one agent problem is also decidable for K45 accessibility relations. Yu et al. (2013) also proved that restricting to propositional event models yields decidability of the epistemic planning problem, even for several agents and arbitrary accessibility relations.

Definition 57. The *propositional epistemic planning problem* is the restriction of the epistemic planning problem to instances with only propositional event models.

Theorem 21 (Yu et al. (2013))**.** *The propositional epistemic planning problem is decidable.*

Before presenting our notion of epistemic protocol synthesis, we illustrate the techniques involved by giving an alternative proof of Theorem 21.

We start with an alternative definition of the epistemic planning problem which we believe may be closer to the intuition and easier to work with. The conception we have of the planning problem, which is also the vision we adopt for our notion of epistemic protocols in the next section, is the following. When one tries to reach some objective, one chooses some actions to perform, which triggers events. However, one does not choose the way these events are perceived by the different agents involved. This is determined by the nature of these events and the observational capabilities of the agents. Therefore, we think that the original formulation of the epistemic planning problem may be somehow

misleading. Indeed, it does not ask to find a sequence of events, but a sequence of pointed event models, thus indicating the way events are perceived. We agree that this is only a matter of technical presentation, as shows Lemma 23 below, but we find it convenient and intuitive to present the problem as follows. There is a single event model, which contains all the possible events. This model may have several unconnected components. We generate the ETL model that contains all possible sequences of events from every possible initial world. The relations $\rightsquigarrow_i$ for each agent i in this model are determined by the accessibility relations in the single event model. In this "full" generated ETL model, that we call the universe, we look for a sequence of events (or a protocol in the next section) that verifies some desired property.

Note that instead of working with pointed event models as is usually done in works on DEL, Wang and Aucher (2013) also consider a "universal" event model. However, this work does not study epistemic planning but rather axiomatizations for DEL.

We now define the one-model epistemic planning problem and show that it captures the standard epistemic planning problem.

Definition 58 (One-model epistemic planning problem)**.** Given a pointed epistemic model $(\mathcal{M}_\iota, w_\iota)$, an event model $\mathcal{E}$, a set of events $\mathsf{E}' \subseteq \mathcal{E}$ and a goal formula Φ, decide if there exists a finite series of events $e_1 \dots e_n$ in E' such that $(\mathcal{M}_\iota, w_\iota) \otimes (\mathcal{E}, e_1) \otimes \dots \otimes (\mathcal{E}, e_n) \models \Phi$. The *propositional one-model epistemic planning problem* is the restriction of the one-model epistemic planning problem to propositional event models.

For an instance $((\mathcal{M}_\iota, w_\iota), \mathcal{E}, \mathsf{E}', \Phi)$ of the one model epistemic problem we define its size as $|((\mathcal{M}_\iota, w_\iota), \mathcal{E}, \mathsf{E}', \Phi)| = |\mathcal{M}_\iota| + |\mathcal{E}| + |\mathsf{E}'| + |\Phi| + |AP|$.

Lemma 23. *The (propositional) epistemic planning problem is linearly reducible to the (propositional) one-model epistemic planning problem.*

Proof. Let $((\mathcal{M}_\iota, w_\iota), \mathfrak{E}, \Phi)$ be an instance of the epistemic planning problem, and note $\mathfrak{E} = \{(\mathcal{E}_1, e_1), \dots, (\mathcal{E}_n, e_n)\}$. Without loss of generality one can assume that for all i, j, $\mathsf{E}_i \cap \mathsf{E}_j = \emptyset$. Define the event model $\mathcal{E} = \bigcup_{i=1}^n \mathcal{E}_i$ and let $\mathsf{E}' = \{e_1, \dots, e_n\}$. $((\mathcal{M}_\iota, w_\iota), \mathcal{E}, \mathsf{E}', \Phi)$ is an instance of the one model epistemic planning problem that has the same size as the original one. It remains to prove that there is a solution in the original instance if, and only if, there is one in the latter instance. Because all the event models are disjoint, an event e in $\mathcal{E}$ belongs to a unique $\mathcal{E}_i$ and is only related to events from $\mathcal{E}_i$. Therefore, for every pointed epistemic model $(\mathcal{M}, w)$ and every pointed event model $(\mathcal{E}_i, e_i) \in \mathfrak{E}$, $(\mathcal{M}, w) \otimes (\mathcal{E}_i, e_i)$ and $(\mathcal{M}, w) \otimes (\mathcal{E}, e_i)$ are bisimilar, and therefore verify the same epistemic formulas – see *e.g.* Blackburn et al. (2006); the result follows.

Finally, observe that if all the event models in $\mathfrak{E}$ are propositional event models, then so is $\mathcal{E}$. □

We fix an instance $((\mathcal{M}_\iota, w_\iota), \mathcal{E}, \mathsf{E}', \Phi)$ of the one-model epistemic planning problem, and we rephrase it in terms of epistemic temporal logic.

Let $\mathcal{H}$ be the ETL model $\mathcal{M}_\iota\mathcal{E}^*$. Recall that we see an ETL model as a labelled forest which plays the role of the universe in the semantics for $m\mathcal{L}_{\rightsquigarrow}$. The semantics also requires a tree in the forest, and because we are looking for a plan/history that starts in w_ι, we simply take the tree $\mathcal{H}_{w_\iota}$, rooted at w_ι, of all histories in $\mathcal{H}$ that start in the world w_ι.

We want to express the existence of a plan in $m\mathcal{L}_{\rightsquigarrow}$. The first step is to translate the $\mathcal{L}^{EL}$ formula Φ into $m\mathcal{L}_{\rightsquigarrow}$, which is done by taking the formula $\widetilde{\Phi}$ (Definition 55). Intuitively, we want to say that there exists a branch in $\mathcal{H}_{w_\iota}$ such that, after a finite amount of time, the formula $\widetilde{\Phi}$ holds, which can be captured by the formula $\mathbf{EF}\widetilde{\Phi}$. However we must ensure that along this branch, only "allowed" events from E' occur before a situation satisfying $\widetilde{\Phi}$ is reached (what happens afterwards is irrelevant). We thus define the $m\mathcal{L}_{\rightsquigarrow}$ formula

$$\texttt{ThereIsAPlan}(\Phi) = \mathbf{E}(p_{w_\iota} \vee \bigvee_{e\in\mathsf{E}'} p_e)\mathbf{U}(\widetilde{\Phi} \wedge (p_{w_\iota} \vee \bigvee_{e\in\mathsf{E}'} p_e)). \tag{7.1}$$

This formula's intended meaning is that either $\widetilde{\Phi}$ already holds in the initial pointed epistemic model, or there is a branch in the tree $\mathcal{H}_{w_\iota}$ such that at some point $\widetilde{\Phi}$ holds, and up to this point all the events that occur in the branch are from E' (except for the first one which is w_ι).

Lemma 24. *The one model epistemic planning problem* $((\mathcal{M}_\iota, w_\iota), \mathcal{E}, \mathsf{E}', \Phi)$ *has a solution if, and only if,* $\mathcal{H}_{w_\iota}, w_\iota \models \texttt{ThereIsAPlan}(\Phi)$.

Proof. We start with the left-to-right implication. Suppose that there is a plan $e_1 \dots e_n$ such that $(\mathcal{M}_\iota, w_\iota) \otimes (\mathcal{E}, e_1) \otimes \dots \otimes (\mathcal{E}, e_n) \models \Phi$ (n can be 0).

Clearly, the history $h = w_\iota e_1 \dots e_n$ is in $\mathcal{H} = \mathcal{M}_\iota\mathcal{E}^*$, and because of Remark 14, there is an infinite branch λ in $\mathcal{H}_{w_\iota}$ such that $h \preccurlyeq \lambda$. We prove that $\mathcal{H}_{w_\iota}, \lambda \models (p_{w_\iota} \vee \bigvee_{e\in\mathsf{E}'} p_e)\mathbf{U}(\widetilde{\Phi} \wedge (p_{w_\iota} \vee \bigvee_{e\in\mathsf{E}'} p_e))$. Because $\lambda[0] = w_\iota$, it holds that $\mathcal{H}_{w_\iota}, \lambda[0] \models p_{w_\iota}$; also, for each i such that $1 \leq i \leq n$, $\lambda[i] = e_i$, hence $\mathcal{H}_{w_\iota}, \lambda^i \models p_{e_i}$ (Lemma 19). Because for every $i \in \{1, \dots, n\}$, $e_i \in \mathsf{E}'$, we have that $\mathcal{H}_{w_\iota}, \lambda^i \models \bigvee_{e\in\mathsf{E}'} p_e$. Also, because $(\mathcal{M}_\iota, w_\iota) \otimes (\mathcal{E}, e_1) \otimes \dots \otimes (\mathcal{E}, e_n) \models \Phi$, by Lemma 20 we have that $\mathcal{H}_{w_\iota}, w_\iota e_1 \dots e_n \models \widetilde{\Phi}$, *i.e.* $\mathcal{H}_{w_\iota}, \lambda^n \models \widetilde{\Phi}$, which concludes.

Now for the right-to-left implication, assume that $\mathcal{H}_{w_\iota}, w_\iota \models \texttt{ThereIsAPlan}(\Phi)$. There is an infinite branch $\lambda \in \mathcal{H}_{w_\iota}$ such that $\mathcal{H}_{w_\iota}, \lambda \models (p_{w_\iota} \vee \bigvee_{e\in\mathsf{E}'} p_e)\mathbf{U}(\widetilde{\Phi} \wedge (p_{w_\iota} \vee \bigvee_{e\in\mathsf{E}'} p_e))$. Therefore there exists $n \geq 0$ such that $\mathcal{H}_{w_\iota}, \lambda^n \models \widetilde{\Phi}$, and for every $i \in \{0, \dots, n\}$, $\mathcal{H}_{w_\iota}, \lambda^i \models p_{w_\iota} \vee \bigvee_{e\in\mathsf{E}'} p_e$. Note $\lambda^n = w_\iota e_1 \dots e_n$. By Lemma 20, $\mathcal{H}_{w_\iota}, \lambda^n \models \widetilde{\Phi}$ implies that $(\mathcal{M}_\iota, w_\iota) \otimes (\mathcal{E}, e_1) \otimes \dots \otimes (\mathcal{E}, e_n) \models \Phi$, and by Lemma 19, $e_i \in \mathsf{E}'$ for every $i \in \{1, \dots, n\}$. Finally, $e_1 \dots e_n$ is a solution plan. □

Now that we have reformulated the one model epistemic planning problem in ETL, we can invoke the fact that ETL models generated by propositional DEL models are regular structures to obtain an equivalent fully-uniform strategy problem. This gives us an alternative proof of Theorem 21.

Proposition 28. *Let* $((\mathcal{M}_\iota, w_\iota), \mathfrak{E}, \Phi)$ *be an instance of the propositional epistemic planning problem. There is a* 2^{AP}*-labelled arena* $\mathcal{G}$ *and transducers* $\{T_i\}_{i\in Ag}$ *over* 2^{AP} *such that there is a solution plan if, and only if, Player 1 has a* $(\{[T_i]\}_{i\in Ag}, \texttt{ThereIsAPlan}(\Phi))$*-uniform strategy in* $\mathcal{G}$.

Proof. Let $((\mathcal{M}_\iota, w_\iota), \mathfrak{E}, \Phi)$ be an instance of the propositional epistemic planning problem. By Lemma 23 there is an equivalent propositional one-model epistemic planning problem $((\mathcal{M}_\iota, w_\iota), \mathcal{E}, \mathsf{E}', \Phi)$ of same size. Letting $\mathcal{H} = \mathcal{M}_\iota\mathcal{E}^*$, we have by Lemma 24 that the latter planning problem admits a solution plan if, and only if, $\mathcal{H}_{w_\iota}, w_\iota \models \texttt{ThereIsAPlan}(\Phi)$.

We now reduce the problem of deciding whether $\mathcal{H}_{w_\iota}, w_\iota \models \texttt{ThereIsAPlan}(\Phi)$ to a fully-uniform strategy problem.

Because $\mathcal{H}$ is PDEL-generated, by Proposition 27 there exists a labelled game arena $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$ and transducers $\{T_i\}_{i\in Ag}$ over 2^{AP} such that (we recall Properties 1-6 of Proposition 27):

1. $|\mathcal{G}| = 2^{|AP|} \cdot (|\mathcal{M}| + |\mathcal{E}|)$,
2. every T_i has one state,
3. there is a bijection $f : \mathcal{H} \to Paths_*(V_\iota)$,
4. the restriction of f to $\mathcal{H}_{w_\iota}$ [4] is a bijection between $\mathcal{H}_{w_\iota}$ and $Plays_*$, and
5. for every $(2^{AP}, \Sigma)$-tree $t \subseteq \mathcal{H}$, for every node $x \in t$, for every state formula $\varphi \in m\mathcal{L}_{\rightsquigarrow}$, $t, x \models \varphi$ iff $f(t), f(x) \models \varphi$.
6. Furthermore, if accessibility relations in $\mathcal{M}$ and $\mathcal{E}$ are reflexive, then for each $i \in Ag$, $[T_i]$ verifies No Miracles.

We have that $\mathcal{H}_{w_\iota}, w_\iota \models \texttt{ThereIsAPlan}(\Phi)$ iff $f(\mathcal{H}_{w_\iota}), f(w_\iota) \models \texttt{ThereIsAPlan}(\Phi)$, *i.e.* $Plays_*, v_\iota \models \texttt{ThereIsAPlan}(\Phi)$.

Notice that because $\texttt{ThereIsAPlan}(\Phi)$ is an existential linear-time formula, $Plays_*, v_\iota \models \texttt{ThereIsAPlan}(\Phi)$ just means that there exists a play that verifies $(p_{w_\iota} \vee \bigvee_{e\in\mathsf{E}'} p_e)\mathbf{U}(\widetilde{\Phi} \wedge (p_{w_\iota} \vee \bigvee_{e\in\mathsf{E}'} p_e))$. It remains to notice that Player 1 having no opponent in $\mathcal{G}$, a strategy for her is just an infinite play. Therefore, using the above Properties 3, 4 and 5, one can easily show the following fact, which terminates the proof of Proposition 28:

Fact 6. *The multi-agent planning problem* $((\mathcal{M}_\iota, w_\iota), \mathfrak{E}, \Phi)$ *has a solution if, and only if, Player 1 has a* $(\{[T_i]\}_{i\in Ag}, \texttt{ThereIsAPlan}(\Phi))$*-uniform strategy in* $\mathcal{G}$*.*

□

Because $\texttt{ThereIsAPlan}(\Phi)$ is a $\mathcal{F}m\mathcal{L}_{\rightsquigarrow}$ formula, we have reduced the propositional epistemic planning problem to an instance of $n\mathsf{FUS}$ (or $n\mathsf{FUS}_{\mathrm{K45NM}}$ if accessibility relations are equivalences). By Theorem 13, we derive the decidability of the propositional epistemic planning problem, as in Theorem 21.

Corollary 14. *The propositional epistemic planning problem is decidable.*

Moreover, the arena $\mathcal{G}$ of Proposition 28 is of size exponential in the size of the epistemic planning problem's input ($\mathcal{G} = 2^{|AP|}(|\mathcal{M}| + |\mathcal{E}|)$), and $\texttt{ThereIsAPlan}(\Phi)$ has the same $\boxbslash$-depth and alternation depth as Φ. With these observations, our upper bounds for the fully-uniform strategy problem give us a $\max(2, k+1)$-Exptime upper bound for objective formulas of $\boxbslash$-depth k, and $\max(2, h+1)$-Exptime for objective formulas of alternation depth h when the accessibility relations are equivalences. Also our decision procedures synthesize a uniform strategy which represents a solution plan if any.

However, we can take advantage of the fact that the problem reduces to a degenerate case in which Player 1 plays alone. First, in this case we obtain an Exptime upper bound, instead of 2-Exptime, for formulas of $\boxbslash$-depth zero. Moreover, as shown in Theorem 22 below, the powerset arena obtained in our decision procedure after the full quantifiers'

4. The set of histories that start with w_ι (see Definition 3, page 13).

elimination can be turned into a finite word automaton that accepts exactly the set of solution plans. Therefore, instead of synthesizing a unique solution plan, we manage to synthesize *all the solution plans*, which may be infinitely many.

Theorem 22. *The propositional epistemic planning problem is in* $k+1$-EXPTIME *for formulas of nesting depth* k*. If in addition the accessibility relations are equivalences and the alternation depth is* h*, then the problem is in* $h+1$-EXPTIME*. Moreover, one can build in the same time a finite word automaton* $\mathcal{P}$ *such that* $\mathcal{L}(\mathcal{P})$ *is the set of all solution plans.*

Proof. Let $((\mathcal{M}_\iota, w_\iota), \mathfrak{E}, \Phi)$ be an instance of the propositional epistemic planning problem such that $d(\Phi) \leq k$. By Proposition 28, there is a 2^{AP}-labelled arena $\mathcal{G}$ and transducers $\{T_i\}_{i\in Ag}$ over 2^{AP} such that there is a solution plan if, and only if, Player 1 has a $(\{[T_i]\}_{i\in Ag}, \mathtt{ThereIsAPlan}(\Phi))$-uniform strategy in $\mathcal{G}$. Moreover (see the proof of Proposition 28), $\mathcal{G}$ and transducers $\{T_i\}_{i\in Ag}$ satisfy Properties 1-6 of Proposition 27. Therefore $|\mathcal{G}| = 2^{|AP|} \cdot (|\mathcal{M}| + |\mathcal{E}|)$ and each T_i has one state. Starting from the instance $(\mathcal{G}, \{[T_i]\}_{i\in Ag}, \mathtt{ThereIsAPlan}(\Phi))$ of $\mathcal{F}n\mathcal{L}^k_{\rightsquigarrow}$ and using our powerset construction and $\boxbslash_i$ elimination procedure (see Theorem 13), we build an equivalent CTL^* game $(\widehat{\mathcal{G}}, \widehat{\mathtt{ThereIsAPlan}}(\Phi))$ where $\widehat{\mathcal{G}}$ is of size k-exponential in $|\mathcal{G}| + \sum_{i\in Ag} |T_i| + |\Phi|$. Because $|\mathcal{G}| = 2^{|AP|} \cdot (|\mathcal{M}| + |\mathcal{E}|)$, $\widehat{\mathcal{G}}$ is of size $(k+1)$-exponential in the input of the epistemic planning problem. Recall that $\mathtt{ThereIsAPlan}(\Phi) = \mathbf{E}(p_{w_\iota} \vee \bigvee_{e\in\mathsf{E}'} p_e)\mathbf{U}(\widetilde{\Phi} \wedge (p_{w_\iota} \vee \bigvee_{e\in\mathsf{E}'} p_e))$. We have that $\widehat{\mathtt{ThereIsAPlan}}(\Phi) = \mathbf{E}(p_{w_\iota} \vee \bigvee_{e\in\mathsf{E}'} p_e)\mathbf{U}(\varphi \wedge (p_{w_\iota} \vee \bigvee_{e\in\mathsf{E}'} p_e))$, where $\varphi = \widehat{\widetilde{\Phi}}$ is a propositional formula: in $\widetilde{\Phi}$, knowledge operators K_i of Φ have become $\boxbslash_i$, and in φ formulas of the form $\boxbslash_i\varphi'$ have been recursively replaced by $p_{\boxbslash_i\varphi'}$. Because there is no temporal operator in Φ, φ is purely propositional. Therefore, φ can be evaluated positionally in $\widehat{\mathcal{G}}$. Let $\widehat{\mathcal{G}} = (V, E, V_\iota, v_\iota, \mu)$. We define $V_\varphi = \{v \in V \mid \mu(v) \models \varphi\}$. Recall that there is a natural bijection f between the set of possible histories in $\mathcal{H}$ (*i.e.* the set of possible plans) and the set of finite paths in $\mathcal{G}$, and there is also a bijection between finite paths of our successive powerset constructions (see Lemma 17). It is therefore easy to see that the set of plans that achieve the goal Φ is in bijection with the set of partial plays in $\widehat{\mathcal{G}}$ that reach V_φ. The set of plans that only use "allowed" events from E' is therefore in bijection with the set of partial plays in $\widehat{\mathcal{G}}$ that reach V_φ while remaining in $\bigvee_{e\in\mathsf{E}'} p_e$ (see Remark 16). To obtain the automaton that accepts the set of solution plans, it only remains to cut moves that go through unauthorized events.

Formally, we let $\mathcal{A} = (\mathsf{E}', Q, \delta, s_\iota, F)$, where:

- $Q = V$, $s_\iota = v_\iota$, $F = V_\varphi$, and
- for $v \in Q$ and $e \in \mathsf{E}'$, $\delta(v, e) = v'$ if there is $v' \in V$ such that $v \to v'$ in $\widehat{\mathcal{G}}$ and $p_e \in \mu(v')$.

It is now easy to verify that:

$$\mathcal{L}(\mathcal{A}) = \{e_1 \dots e_n \mid \forall i, e_i \in \mathsf{E}' \text{ and } (\mathcal{M}_\iota, w_\iota) \otimes (\mathcal{E}, e_1) \otimes \dots \otimes (\mathcal{E}, e_n) \models \Phi\}.$$

Automaton $\mathcal{A}$ is built in time linear in the size of $\widehat{\mathcal{G}}$. We remind that $\widehat{\mathcal{G}}$ is of size $(k+1)$-exponential in general (where k is the modal depth of Φ), and in case the accessibility relations are equivalences, it is of size $(h+1)$-exponential where h is the alternation depth of Φ. The upper bounds follow from the fact that testing the emptiness of $\mathcal{L}(\mathcal{A})$ can be done in time linear in the size of $\mathcal{A}$. □

7.3.2 The epistemic protocol synthesis problem

We generalize the notion of epistemic planning in three directions. First, we no longer consider finite sequences of actions but infinite ones. As a consequence, we need not stick to reachability objectives as in planning (where the aim is to reach a state of the world that verifies some formula), and we therefore allow for any epistemic temporal formula as objective, which is the second generalization. Finally, we no longer look for a single series of events, but we try to synthesize a *protocol*, *i.e.* a set of plans.

Epistemic protocols

We define a notion of epistemic protocol, and we discuss related notions of protocols in the literature.

Definition 59. Given an ETL model $\mathcal{H} = (\Sigma, H, \{\rightsquigarrow_i\}_{i \in Ag}, V)$, an *epistemic protocol* is a Σ-forest $P \subseteq H$; it is *rooted* if it is a tree.

The literature of logics for knowledge and time contains several notions of protocols. In the community of epistemic temporal logics, what is called a protocol is often the domain of the model itself (Parikh and Ramanujam, 2003; Pacuit and van Benthem, 2006; Pacuit, 2007; van Benthem et al., 2009), and it represents the set of possible behaviours of some system. This notion is similar to DEL protocols considered for example in van Benthem et al. (2009) and Wang and Aucher (2013), which map each possible world of an initial epistemic model to a forest of events, thus defining a set of possible behaviours in which DEL formulas are evaluated.

In van der Meyden and Vardi (1998) and van der Meyden and Wilke (2005), a problem of protocol synthesis from epistemic temporal specifications is studied, which is very close to the one we consider below. We discussed this work in Section 6.5.2, where we described it in terms of distributed strategy synthesis in a concurrent game structure. With the vocabulary of the ETL community, a concurrent game structure is an *interpreted environment*, modeling how the actions of various agents impact this environment and how these agents observe it, and a *(joint) protocol* is a (distributed) strategy for the agents. The aim is then to synthesize a protocol that verifies some epistemic temporal property. This notion of protocol is different from the former one. Instead of defining the system, it denotes some sub-behavior of the system that verifies a desired property. This is reminiscent of the supervisory control task as defined by Ramadge and Wonham (1987) which, given a system and a safety property aims at pruning the system's behaviour so that the controlled system is safe.

Our notion of protocol is closer to the latter one, though more abstract: it is a subset of the system, that represents some control of, or strategy in the system which aims at achieving some property. However, in all the works above mentioned, it is assumed that the agents have complete information in the sense that they know the protocol. More precisely, given a protocol, all the agents know what can happen in this protocol, and they rule out from their beliefs or knowledge behaviours that do not follow this protocol. We weaken this assumption and consider that some, or even all of the agents may not be aware of the restriction imposed on the system by the protocol; or maybe that they are aware of the existence of such a restriction but do not know what it is exactly, because

it is enforced by someone else for example. Therefore, according to their observations, such agents might very well consider possible some behaviours that are not allowed by the protocol. However they do have complete information about the system itself, so that they still rule out behaviours that are not possible in the original system.

One could claim that it makes little sense for an agent not to know the protocol. Indeed, how can she behave if she does not know what actions are allowed? However we believe that it can make sense for particular modeling purposes, to consider some passive agents that only observe the evolution of the world, and therefore do not need to know what the protocol is. This is the case of DEL agents for example, as discussed before Lemma 20, page 107.

An example where the agent is active but where it still makes sense to assume that she ignores the protocol, is found in games with opacity condition as described in Section 3.4. The system is made of all the possible plays in the game. Fixing a strategy for one of the players restricts the set of possible behaviours of the system, but there is no reason why the other player should know what this restriction is, as in general a player does not know the strategy of her opponent. Note that in this formalism, a strategy for a player does not restrict what the opponent is entitled to do at each point of a play, so that she can play without knowing the strategy/protocol. When imperfect information is involved as in games with opacity condition, and a strategy of Defender is fixed, not only Attacker does not know at a given point of a play what future behaviours are still possible according to Defender's strategy, but also she may not know exactly what has *already* been played: among the partial plays she considers possible, Attacker does not know which ones are ruled out by Defender's strategy. This is why in Section 3.4, to capture the knowledge of Attacker, we used the full quantifier of our logic $\mathcal{L}_{\rightsquigarrow}$, as it represents the knowledge in the *full* system, while the strict quantifier represents the knowledge in the system *restricted* to the protocol.

Epistemic protocol synthesis

We now define our epistemic protocol synthesis problem. Because the models of our language are trees we only consider rooted epistemic protocols. It is not clear how to define the meaning of a forest verifying an $m\mathcal{L}_{\rightsquigarrow}$ formula. Should the formula hold in at least one tree of the forest, in all the trees, or something in between? In the first two cases, the results we establish for the epistemic protocol synthesis problem would also hold for non-rooted protocols. It suffices to turn forests into trees by adding to each forest an artificial root that branches to all the roots of the forest.

Remember that for an ETL model $\mathcal{H} = (H, \{\rightsquigarrow_i\}_{i\in Ag}, V)$, a labelled tree $t \subseteq H$ and a formula $\Phi \in m\mathcal{L}_{\rightsquigarrow}$, $t \models \Phi$ stands for $\{\rightsquigarrow_i\}_{i\in Ag}, \mathcal{U}, t, r \models \Phi$, where $\mathcal{U} = H$ is the universe and r is the root of t (see Section 7.1.2).

Definition 60 (Epistemic protocol synthesis problem)**.** Given an initial pointed epistemic model $(\mathcal{M}_\iota, w_\iota)$, an event model $\mathcal{E}$ and an $m\mathcal{L}_{\rightsquigarrow}$ formula Φ, decide if there is an epistemic protocol $P \subseteq \mathcal{M}_\iota\mathcal{E}^*$ rooted in w_ι such that $P \models \Phi$.

The size of an instance $((\mathcal{M}_\iota, w_\iota), \mathcal{E}, \Phi)$ of the epistemic protocol synthesis problem is $|((\mathcal{M}_\iota, w_\iota), \mathcal{E}, \Phi)| = |\mathcal{M}_\iota| + |\mathcal{E}| + |\Phi| + |AP|$.

Remark 18. Note that we consider a unique event model. We could define the problem with several event models, but it would make the definition more cumbersome without really adding anything, as justified in Section 7.3.1. Also, in the definition of the problem, we allow for arbitrary $m\mathcal{L}_{\rightsquigarrow}$ formulas, and in particular we allow for both strict and full quantifiers for each agent. We want to stress out that here, using a full quantifier $\boxbslash_i$ corresponds to the assumption that agent i *does not know* the protocol: the set of worlds it considers possible at some point in time is not refined by the knowledge of what sequences of events are allowed by the protocol. On the other hand, capturing the knowledge of an agent who *does* know the protocol can also be done by using the strict quantifier instead.

Proposition 29. *The epistemic protocol synthesis problem is undecidable.*

Proof. The epistemic planning problem, which is undecidable, is clearly subsumed by the epistemic protocol synthesis problem. □

We now identify some restrictions that make the epistemic protocol synthesis problem decidable. First we restrict to propositional event models, which enables us to reduce to a uniform strategy problem; we make two more assumptions that ensure that the obtained uniform strategy problem problem is an instance of $n\mathsf{SFUS}$, which is decidable. The first assumption is that no strict quantifier can occur in the scope of a full quantifier (the formula is in $\mathcal{SF}n\mathcal{L}_{\rightsquigarrow}$). The second assumption is that strict quantifiers must rely on recognizable relations. In Definition 54, the definition of the relations in a DEL-generated model rely on the assumption that the agents have (synchronous) perfect recall. This kind of relation is not recognizable in general, reason why we discuss a notion of agent with bounded memory.

Bounded memory

Several notions of agents with bounded memory have been studied in the literature. A first approach is to consider that the agent's memory is a fixed-size "window" that translates along a history. For example, in the DEL framework, Liu (2009) considers a notion of agent with bounded memory who remembers only the last k events. We consider a more general notion of *memory structure*, as studied for example in Dziembowski et al. (1997) to study the memory required to win games with ω-regular winning conditions.

Let $\mathcal{M} = (W, \{R_i\}_{1\leq i\leq n}, V)$ be an epistemic model, and let $\mathcal{E} = (\mathsf{E}, \{\mathsf{R}_i\}_{1\leq i\leq n}, \mathrm{pre}, \mathrm{post})$ be an event model. Consider an agent $i \in Ag$ and assume that all R_i and all R_i are equivalence relations. This assumption enables us to define a meaningful notion of *bounded memory observation relation*. We say that agent $i \in Ag$ has *bounded memory* if there is a finite memory structure $\mathfrak{M}_i = (W \cup \mathsf{E}, M, \delta, m_\iota)$ – see Definition 4 – that represents her memory. To remain consistent with the observational power of the agent, we assume that "indistinguishable" events affect the memory of the agent in the same manner. Formally, for $m \in M$ and $w, u \in W$ such that $w\, R_i\, u$, we require that $\delta(m, w) = \delta(m, u)$. Similarly, if $e, f \in \mathsf{E}$ and $e\, \mathsf{R}_i\, f$, then $\delta(m, e) = \delta(m, f)$.

Definition 61 (Bounded memory observation relation)**.** Suppose agent $i \in Ag$ has bounded memory represented by $\mathfrak{M}_i = (W \cup \mathsf{E}, M, \delta, m_\iota)$. We define the *bounded memory observation relation* $\sim_{\mathfrak{M}_i}$ over $(W \cup \mathsf{E})^*$ as follows:

For $h, h' \in (W \cup \mathsf{E})^*$, $h \sim_{\mathfrak{M}_i} h'$ if $\delta(m_\iota, h) = \delta(m_\iota, h')$.

Observe that, as expected, if the agent cannot distinguish two histories with perfect recall, neither can she with bounded memory. Indeed, thanks to the assumption that indistinguishable events trigger the same transitions in the memory structure, we have: $\rightsquigarrow_i \subseteq \rightsquigarrow_{\mathfrak{M}_i}$. Also, $\rightsquigarrow_{\mathfrak{M}_i}$ is coarser than $\rightsquigarrow_i$ in general, as histories that are not $\rightsquigarrow_i$-related can lead to the same memory state.

We now prove that $\rightsquigarrow_{\mathfrak{M}_i}$ is a recognizable relation.

Lemma 25. *Suppose agent $i \in Ag$ has bounded memory represented by $\mathfrak{M}_i$. The bounded memory observation relation $\rightsquigarrow_{\mathfrak{M}_i}$ is recognizable. In addition, one can effectively build from $\mathfrak{M}_i$ a deterministic word automaton $\mathcal{B}_i$ that recognizes $\rightsquigarrow_{\mathfrak{M}_i}$, and the size of $\mathcal{B}_i$ is exponential in the size of $\mathfrak{M}_i$.*

Proof. Let $\mathfrak{M}_i = (W \cup \mathsf{E}, M, \delta, m_\iota)$. For each $m \in M$, we define the language $\mathcal{L}_m = \{h \in (W \cup \mathsf{E})^* \mid \delta(m_\iota, h) = m\}$. $\mathcal{L}_m$ is clearly a regular language as it is accepted by the finite state automaton $\mathcal{A}_m = (W \cup \mathsf{E}, M, \delta, m_\iota, F_m = \{m\})$. We have $\rightsquigarrow_{\mathfrak{M}_i} = \bigcup_{m \in M} \mathcal{L}_m \times \mathcal{L}_m$, therefore $\rightsquigarrow_{\mathfrak{M}_i}$ is a recognizable relation (see Definition 15).

To build a recognizer from the memory structure $\mathfrak{M}_i$, we combine $\mathfrak{M}_i$ with $\overline{\mathfrak{M}_i}$, which is the same machine with reversed transitions (hence it is nondeterministic in general). From $\mathfrak{M}_i$ and $\overline{\mathfrak{M}_i}$, building a nondeterministic automaton over alphabet $W \cup \mathsf{E} \cup \{\#\}$ and of size $O(|\mathfrak{M}_i|^2)$ that recognizes $\{h\#h' \mid h \rightsquigarrow_{\mathfrak{M}_i} h'\}$ is an easy task. Determinizing this word automaton yields a recognizer $\mathcal{B}_i$ for $\rightsquigarrow_{\mathfrak{M}_i}$, whose size is exponential in $|\mathfrak{M}_i|$. □

From now on, given an initial epistemic model $\mathcal{M}$ and an event model $\mathcal{E}$, if agent i has bounded memory represented by $\mathfrak{M}_i$, then $\rightsquigarrow_i$ is replaced in $\mathcal{M}\mathcal{E}^*$ by $\rightsquigarrow_{\mathfrak{M}_i}$, and formulas of the form $\boxminus_i\varphi$ and $\boxplus_i\varphi$ are therefore evaluated with regard to $\rightsquigarrow_{\mathfrak{M}_i}$.

Definition 62. Let $\mathcal{H} = \mathcal{M}\mathcal{E}^* = \{W \cup \mathsf{E}, H, \{\rightsquigarrow_i\}_{i \in Ag}, V\}$. Suppose that the set of agents having bounded memory is $Ag_{bm} \subseteq Ag$, and the memory of each agent $i \in Ag_{bm}$ is represented by $\mathfrak{M}_i$. Then for a 2^{AP}-labelled tree $t \subseteq \mathcal{H}$, a node $x \in t$ and a formula $\Phi \in m\mathcal{L}_\sim$, we let $t, x \models \varphi$ mean that $\{\sim'_i\}_{i \in Ag}, \mathcal{U}, t, x \models \varphi$, where $\mathcal{U} = H$ is the universe, $\rightsquigarrow'_i = \rightsquigarrow_{\mathfrak{M}_i}$ if $i \in Ag_{bm}$, and $\rightsquigarrow'_i = \rightsquigarrow_i$ otherwise.

A decidable epistemic protocol synthesis problem

We identify some restrictions on the epistemic protocol synthesis problem that ensure decidability. More precisely, we show that when:

1. the events are propositional,
2. the agents who "know" the protocol have bounded memory, and their memory structure is known,
3. the agents who "do not know" the protocol cannot reason about the knowledge of agents who do,

then the epistemic protocol synthesis problem is decidable. We motivate these assumptions with the following (semi)-informal discussion.

As already explained in Proposition 27, Assumption 1 enables us to reduce the problem to a uniform strategy problem. Informally, restricting to propositional event models means that the conditions for an event to occur cannot depend on the knowledge state of agents,

but only on the facts of the world. This reduces the complexity of the events we can model, but the class of propositional event models still captures many interesting kinds of events. For example, the event models in Figure 7.2 and Figure 7.3 are propositional.

For Assumption 2, recall the discussion following our Definition 59 of protocols. When we say that an agent "knows" the protocol, we mean more precisely that the semantics of its knowledge is the one of the strict quantifier, *i.e.* she does not consider possible histories outside the protocol. On the other hand, an agent who "does not know" the protocol is an agent whose knowledge has the semantics of the full quantifier. As it is well-known, letting agents who "know" the protocol have perfect-recall leads to undecidability as soon as at least two agents are involved – see van der Meyden and Wilke (2005) for example. However, we have seen that the strict quantifier can be managed if it is associated with recognizable relations. As Lemma 25 illustrates, the observational ability of agents with bounded memory can be represented by recognizable relations. Even though it is often assumed that agents have perfect recall, for example to model the "most powerful" adversary, in practice all agents we can think of have a bounded memory, even though it may be very complex to model (especially for humans). Imagine a situation with a team A of agents (the nice ones) and a team B (the bad ones). Team A aims at designing a joint protocol so that some property holds, regardless of the behaviour of team B. It is not too unreasonable to assume that the agents in team A know their memory structure. Indeed, a robotic agent for example can possess the information of how its memory is implemented, and it can share it with other members of the team in order to compute the protocol. Also, assuming that all the agents in the team can be trusted and that no one reveals the protocol to team B, we can assume that only the agents in team A will "know" the protocol when executed. Note that undermining this assumption is the strong other assumption that team B does not have the capability to guess the protocol by themselves. However there is in general no unicity of a solution protocol, which makes it unlikely for team B to "guess" which protocol is used by team A. To sum up, this kind of situation verifies the second assumption.

We turn to Assumption 3. For agents who "do not know" the protocol, our results on the fully-uniform strategy problem show that we can manage any rational relations for the semantics of the full quantifier. However, the only case where we know how to manage both agents knowing the protocol – strict quantifiers – and agents ignoring the protocol – full quantifiers – is when there is no strict quantifier in the scope of a full quantifier (see Section 6.4). This can be interpreted as the fact that no agent who ignores the protocol can reason about the knowledge of an agent who does. We argue that this seems realistic. Assume that it is common knowledge how each agent observes the world. Assume also that agent i knows the protocol, agent j ignores the protocol and agent i knows that it is so. Agent i can reason about the knowledge of agent j by "simulating" not to know the protocol: when agent i evaluates the knowledge of agent j, she can consider $\rightsquigarrow_j$-related histories [5] that are outside the protocol, even though she knows that they are not possible. However, even if agent j knew that agent i knows the protocol, she could not "simulate" the knowledge of the protocol without knowing the protocol itself. Therefore it seems unrealistic to let agent j reason about the knowledge of agent i.

We now define the problem formally.

5. we have assumed that the agents' observational abilities are common knowledge.

Definition 63 (Bounded memory epistemic protocol synthesis problem (BMEPS))**.** Given an initial pointed epistemic model $(\mathcal{M}_\iota, w_\iota)$, a propositional event model $\mathcal{E}$, a subset of agents $Ag_{bm} \subseteq Ag$ with a memory structure $\mathfrak{M}_i$ for each $i \in Ag_{bm}$, and an $\mathcal{SF}m\mathcal{L}_{\rightsquigarrow}$-formula Φ such that, if $\boxminus_i\varphi \in Sub(\Phi)$ (resp. $\boxplus_i\varphi \in Sub(\Phi)$), then $i \in Ag_{bm}$ (resp. $i \in Ag \setminus Ag_{bm}$), decide if there is an epistemic protocol $P \subseteq \mathcal{M}_\iota\mathcal{E}^*$ rooted in w_ι such that $P \models \Phi$.

The size of a BMEPS instance is defined as follows: $|(\mathcal{M}, w_\iota), \mathcal{E}, \{\mathfrak{M}_i\}_{i\in Ag_{bm}}, \Phi| = |\mathcal{M}| + |\mathcal{E}| + \sum_{i\in Ag_{bm}} |\mathfrak{M}_i| + |\Phi| + |AP|$.

Theorem 23. *BMEPS is decidable. If the $\boxplus$-depth of the goal formulas is bounded by k, then the problem is in* $\max(2, k+1)$-Exptime. *If, in addition, the accessibility relations are equivalence relations and the alternation depth of the goal formulas is bounded by h, then the problem is in* $\max(2, h+1)$-Exptime.

Proof. Let $((\mathcal{M}_\iota, w_\iota), \mathcal{E}, \{\mathfrak{M}_i\}_{i\in Ag_{bm}}, \Phi)$ be an instance of BMEPS, where $Ag_{bm} \subseteq Ag$ is the set of agents who have bounded memory. Let $\mathcal{H} = \mathcal{M}_\iota\mathcal{E}^*$. By Proposition 27, there is a labelled game arena $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$ and transducers $\{T_i\}_{i\in Ag}$ over 2^{AP} such that:

1. $|\mathcal{G}| = 2^{|AP|} \cdot (|\mathcal{M}| + |\mathcal{E}|)$,
2. every T_i has one state,
3. there is a bijection $f : \mathcal{H} \to Paths_*(V_\iota)$,
4. the restriction of f to $\mathcal{H}_{w_\iota}$ is a bijection between $\mathcal{H}_{w_\iota}$ and $Plays_*$, and
5. for every $(2^{AP}, \Sigma)$-tree $t \subseteq \mathcal{H}$, for every node $x \in t$, for every state formula $\varphi \in m\mathcal{L}_{\rightsquigarrow}$, $t, x \models \varphi$ iff $f(t), f(x) \models \varphi$.
6. Furthermore, if accessibility relations in $\mathcal{M}$ and $\mathcal{E}$ are reflexive, then for each $i \in Ag$, $[T_i]$ satisfies No Miracles.

First, build $\mathcal{B}_i$ from $\mathfrak{M}_i$ for each $i \in Ag_{bm}$, as described in Lemma 25. Observe that Point 5 still holds when we consider the semantics for agents with bounded memory, as we just replace relation $[T_i]$ with $\rightsquigarrow_{\mathfrak{M}_i}$ for $i \in Ag_{bm}$. Next, because Player 1 owns all the positions in $\mathcal{G}$, there is a one-to-one correspondence between the set of possible protocols rooted in w_ι and generalized strategies for Player 1 in $\mathcal{G}$. The problem therefore reduces to the existence of a $(\{\rightsquigarrow'_i\}_{i\in Ag}, \Phi)$-generalized uniform strategy for Player 1 in $\mathcal{G}$, where $\rightsquigarrow'_i = \rightsquigarrow_{\mathfrak{M}_i}$ if $i \in Ag_{bm}$, and $\rightsquigarrow'_i = [T_i]$ otherwise. Observe that $(\mathcal{G}, \{\mathcal{B}_i\}_{i\in Ag_{bm}}, \{T_i\}_{i\notin Ag_{bm}}, \Phi)$ is an instance of mSFUS. By Point 1, $\mathcal{G}$ is of size exponential in $|AP|$, by Point 2 each T_i has one state, and by Lemma 25 each $\mathcal{B}_i$ is of size exponential in $|\mathfrak{M}_i|$. The instance $(\mathcal{G}, \{\mathcal{B}_i\}_{i\in Ag_{bm}}, \{T_i\}_{i\notin Ag_{bm}}, \Phi)$ of mSFUS is therefore of size exponential in the size of the original BMEPS instance $((\mathcal{M}_\iota, w_\iota), \mathcal{E}, \{\mathfrak{M}_i\}_{i\in Ag_{bm}}, \Phi)$. The first part of Theorem 23 then follows from Theorem 15. Concerning the case where in addition accessibility relations in the initial epistemic model and the event model are equivalence relations, we have that for each $i \in Ag$, $[T_i]$ is also an equivalence relation, and by Point 6, $[T_i]$ also satisfies No Miracles. $(\mathcal{G}, \{\mathcal{B}_i\}_{i\in Ag_{bm}}, \{T_i\}_{i\notin Ag_{bm}}, \Phi)$ is therefore an instance of $n\mathsf{SFUS}_{\mathrm{K45NM}}$, and Theorem 16 provides the upper bounds. □

7.4 Conclusion and perspectives

In this chapter we first extended existing results concerning connections between different frameworks for the logical study of knowledge and time. While the matter has been

well studied concerning DEL and ETL, we have established a new bridge between these frameworks and regular structures. This allowed us to apply the techniques developed in this thesis for uniform strategies to the epistemic planning problem in the DEL framework. We obtained an alternative proof of a recent decidability result concerning the case of propositional events, and we also provided accurate upper bounds on the time complexity of this problem. In addition our approach allowed us to synthesize a finite automaton that generates *all* the solution plans for this epistemic planning problem.

Then we considered a problem of epistemic protocol synthesis which generalizes the epistemic planning problem in several regards, by considering infinite trees of events and epistemic temporal specifications. In addition, the two quantifiers of our language $n\mathcal{L}_{\sim}$ allow us to distinguish, in the specification of a protocol, between agents who *know* the protocol and agents who *ignore* it. We proved that if agents who know the protocol have bounded memory, agents who ignore the protocol have rational observation relations, and they cannot reason about the knowledge of agents who do know the protocol, then the epistemic protocol synthesis problem can be decided, and a solution protocol can be synthesized if any.

On the topic of epistemic protocols, a next step would be to apply techniques from control theory and quantified μ-calculus (Riedweg and Pinchinat, 2003) to synthesize *maximal permissive* epistemic protocols. In general such objects only exist for safety objectives, but recently a weaker notion of *permissive strategy* has been studied in the context of parity games (Bernet et al., 2002). A strategy is permissive if it contains the behaviours of all memoryless strategies, and such strategies always exist in parity games. Similar notions may be introduced for protocols with epistemic temporal objectives to capture concepts of "sufficiently permissive" protocols.

Chapter 8

Conclusion and perspectives

In this thesis we have defined and studied in depth a notion of uniform strategies, which are strategies subject to properties involving sets of plays. We summarize the contribution of each chapter and discuss perspectives.

In Chapter 3, we defined the logic $\mathcal{L}_{\rightsquigarrow}$ and used it to specify uniformity properties of strategies. Then, we demonstrated the relevance of our notion by capturing all the motivating examples of strategies with "horizontal" constraints that we described in the introduction of this document. In fact we capture very expressive properties of strategies by allowing for arbitrary binary relations between plays, as long as they are rational. We have also seen that the full quantifier $\boxbslash$ can be used to model the knowledge of a player (or an agent) who *does not know the strategy* played by his opponent. The strict quantifier $\bigcirc\!\!\!\!\!\!\backslash$, however, captures the knowledge of players who *know* the strategy. The question of whether a player knows or not the strategy being played, though crucial when interpreting knowledge in strategic situations, has surprisingly received very little attention until now. We believe that studying in detail the properties of our strict and full quantifiers may be an interesting first approach to formalize and study the matter of "who knows who does what".

In Chapter 4, we first proved that the existence of uniform strategies for uniformity properties involving only strict quantifiers is undecidable for the class of regular equivalence relations. We defined jumping alternating tree automata, that extend alternating tree automata by allowing for jumps between related nodes of the input tree. We established that, when restricting to recognizable relations, jumping tree automata can be simulated by two-way tree automata. From this we obtained that the strictly-uniform strategy problem is decidable for recognizable relations, and we proved that it is 2-Exptime-complete. In addition, the automata techniques that we use allow us to synthesize a uniform strategy whenever there exists one. We believe that the notion of jumping tree automata may be a relevant theoretical tool to study logics with knowledge. For example, it seems that jumping tree automata may be a good candidate for an automata-theoretic counterpart to the μ-calculus with knowledge. It would also be interesting to try to identify a subclass of rational relations that contains synchronous and asynchronous perfect recall relations, and for which the strictly uniform strategy problem would still be decidable. This may

give sufficient conditions on the observational abilities of players for two-player games with imperfect information and perfect recall to be decidable.

In Chapter 5, we proved that the existence of uniform strategies for uniformity properties that only use the full quantifier can be decided for the whole class of rational relations. Our decision procedure relies on a notion of information set automaton that computes information sets for rational relations. We also established that for rational relations that verify transitivity, Euclideanity and No Miracles, the uniform strategy problem is only 2-EXPTIME-complete. To establish this result we use a notion of information set bisimulation in information set automata. A direction that we believe deserves investigation concerns these information set automata. Because they allow for the computation of information sets, they may provide a way to define a general powerset construction for games with imperfect information and some subclass of rational relations. The two sorts of relations (apart from players with bounded memory) for which a powerset construction is known are synchronous and asynchronous perfect-recall (Reif, 1984; Chatterjee et al., 2006; Puchala, 2010). Identifying a class of relations that strictly contains these relations and for which a powerset construction exists would be interesting as it would be a second way to obtain insights on what characteristics of players make these games decidable. This may also yield results concerning games with imperfect recall but unbounded memory, the study of which has recently received renewed attention (Berwanger et al., 2012).

In Chapter 6, we generalized our framework to manage several relations between plays in a game. We established that our results from Chapters 4 and 5 still hold in this extended setting. The only case for which the results are not the same is the one of fully-uniform strategies with K45NM relations, for which switching to several relations raises the complexity of the uniform strategy problem from 2-EXPTIME to nonelementary. We then proved that techniques from Chapter 4 and 5 can be combined to solve the uniform strategy problem for a class of properties involving both strict and full quantifiers. Our contribution provides a unified proof of decidability for the model-checking of epistemic temporal logics with rational relations. We also described how, using uniformity properties with several strict quantifiers, we can reduce the problem of solving a concurrent game with n players and imperfect information to solving a uniform strategy problem in a two-player game arena. Our contribution shows that we can solve such games with epistemic temporal objectives if the agents have bounded memory, even if we want the knowledge of the players to be restricted to the outcomes of the distributed strategy – which forbids powerset construction techniques. A possible extension of the framework would be in the direction of strategic logics, with quantifiers over uniform strategies. It would also be interesting to consider a language that allows the specification at the syntactic level of the relations attached to each quantifier.

In Chapter 7, we first extended existing results concerning connections between different frameworks for the logical study of knowledge and time. While the matter has been well studied concerning DEL and ETL, we have established a new bridge between these frameworks and regular structures. This allowed us to apply the techniques developed in this thesis for uniform strategies to the epistemic planning problem in the DEL frame-

work. We obtained an alternative proof of a recent decidability result concerning the case of propositional events, and in addition our approach allowed us to synthesize a finite automaton that generates all the solution plans. Then we considered a problem of epistemic protocol synthesis which generalizes the epistemic planning problem in several regards, by considering infinite trees of events and epistemic temporal specifications. In addition, the two quantifiers of our language $n\mathcal{L}_{\rightsquigarrow}$ allow us to distinguish, in the specification of a protocol, between agents who know the protocol and agents who ignore it. We proved that when the agents who know the protocol have bounded memory, and when agents who ignore the protocol cannot reason about the knowledge of agents who do know the protocol, then the epistemic protocol synthesis problem can be decided, and a solution protocol can be synthesized if any. On the topic of epistemic protocols, a next step would be to apply techniques from control theory and quantified μ-calculus (Riedweg and Pinchinat, 2003) to synthesize *maximal permissive* epistemic protocols. In general such objects only exist for safety objectives, but recently a weaker notion of *permissive strategy* has been studied in the context of parity games (Bernet et al., 2002). A strategy is permissive if it contains the behaviours of all memoryless strategies, and such strategies always exist in parity games. Similar notions may be introduced for protocols with epistemic temporal objectives to capture concepts of "sufficiently permissive" protocols.

Appendix A

Proof of Proposition 20

Proposition 20. *For $k \in \mathbb{N}$, FUS_{k+1} is $(k+1)$-*Exptime*-hard even if the $\mathcal{FL}^{k+1}_{\rightsquigarrow}$ formula is assumed to be fixed and the transducer is assumed to be synchronous.*

For each $k \in \mathbb{N}$, let $\exp[k]$ denote the class of functions $f : \mathbb{N} \to \mathbb{N}$ such that for some constant $c \geq 1$, $f(n) = \exp^k(n^c)$ for all $n \in \mathbb{N}$.

Fix $k \geq 0$. Proposition 20 is proved by a polynomial-time reduction from the word problem for $\exp[k]$-space bounded *alternating* Turing Machines with a binary branching degree and without halting configurations, which is a well-known $(k+1)$-Exptime-complete problem (Chandra et al., 1981). Fix such an alternating Turing Machine $\mathcal{M} = (A, Q = Q_\exists \cup Q_\forall, q_\iota, \delta, F)$ over the input alphabet A, where the set of states Q is partitioned into a set $Q_\exists$ of existential states and a set $Q_\forall$ of universal states, q_ι is the initial state, F is the set of accepting states, and the transition function is of type

$$\delta : Q \times A \to (Q \times A \times \{\leftarrow, \rightarrow\}) \times (Q \times A \times \{\leftarrow, \rightarrow\})$$

(in each step, $\mathcal{M}$ overwrites the tape cell being scanned, and the tape head moves one position to the left – $\leftarrow$ – or to the right – $\rightarrow$). Fix an input α and let $n = |\alpha|$.

Note that a configuration of $\mathcal{M}$, or *TM configuration*, can be seen as a word $\alpha_1 \cdot (q, a) \cdot \alpha_2$ in $A^* \cdot (Q \times A) \cdot A^*$, where $\alpha_1 \cdot a \cdot \alpha_2$ denotes the tape content, q the current state, and the reading head is at position $|\alpha_1| + 1$. A TM configuration is *well-formed* if it has length exactly $\exp^k(n)$. Since $\mathcal{M}$ is $\exp[k]$-space bounded, without loss of generality, we can assume that each reachable TM configuration from the fixed input α is well-formed. In particular, the *initial TM configuration* is the unique well-formed TM configuration having the form $(q_\iota, \alpha(1))\, \alpha(2) \ldots \alpha(n) \# \ldots \#$, where $\#$ is the blank symbol. For a TM configuration $C = \alpha_1 \cdot (q, a) \cdot \alpha_2$, the *left* (resp., *right*) successor of C is the successor of C obtained by choosing the left (resp., the right) triple in $\delta(q, a)$. A *computation tree of* $\mathcal{M}$ (over α) is an infinite binary tree whose nodes are labelled by *well-formed* TM configurations and that verifies: (i) the root is labelled by the initial TM configuration, (ii) each node labelled by an *existential* TM configuration C (i.e., the associated state is in $Q_\exists$) has a unique child, labelled by some successor of C, and (iii) each node labelled by an *universal* TM configuration C (i.e., the associated state is in $Q_\forall$) has two children, labelled by the left and right successors of C, respectively. A computation tree of $\mathcal{M}$ is accepting if each infinite branch from the root visits some accepting TM configuration. $\mathcal{M}$ accepts α *iff* there is an accepting computation tree of $\mathcal{M}$.

Proposition 20 directly follows from the following proposition.

Proposition 30. *There is a* fixed $\mathcal{FL}^{k+1}_{\sim}$ *formula* φ *(independent of* n *and* $\mathcal{M}$*) such that one can build – in time polynomial in* n *and the size of* $\mathcal{M}$ *– a* 2^{AP}*-labelled arena* $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$ *and a finite-state synchronous transducer* T *over the alphabet* 2^{AP} *so that* $\mathcal{M}$ *accepts* α iff *Player 1 in* $\mathcal{G}$ *admits a* $([T], \varphi)$*-fully-uniform strategy.*

Proof of Proposition 30 We assume that $k \geq 1$ (the case $k = 0$ being simpler). First, we define a suitable encoding of (well-formed) TM configurations, obtained by using the following set AP of atomic propositions:

$$AP := A \cup (Q \times A) \cup \{\$_1, \ldots, \$_k, \$, \$_{acc}, 0, 1, L, R, \exists, \forall\} \cup \{inc, =, good\}$$

For each cell of a well-formed TM configuration C, we keep track of the content of the cell together with a suitable encoding of the cell number which is a natural number in $[0, \exp^k(n) - 1]$. Thus, for all $1 \leq h \leq k$, we define the notions of *h-block* and *well-formed h-block*. Essentially, for $h < k$, well-formed h-blocks are finite words over $\{\$_1, \ldots, \$_h, 0, 1\}$ which encode integers in $[0, \exp^h(n) - 1]$, while well-formed k-blocks are finite words over $A \cup (Q \times A) \cup \{\$_1, \ldots, \$_k, 0, 1\}$ which encode the cells of well-formed TM configurations. In particular, for $h > 1$, a well-formed h-block encoding a natural number $m \in [0, \exp^h(n) - 1]$ is a sequence of $\exp^{h-1}(n)$ $(h-1)$-blocks, where the i^{th} $(h-1)$-block encodes both the value and (recursively) the position of the i^{th}-bit in the binary representation of m. Formally, the set of (well-formed) h-blocks is defined by induction on h as follows:

Base Step: $h = 1$. The notions of 1-block and well-formed 1-block coincide, and a 1-block is a finite word bl having the form $bl = \$_1 \tau bit_1 \ldots bit_n \$_1$ such that $bit_1, \ldots, bit_n \in \{0, 1\}$ and $\tau \in \{0, 1\}$ if $1 < k$, and $\tau \in A \cup (Q \times A)$ otherwise. We say that bit_i (for $1 \leq i \leq n$) is the i^{th} bit of bl. The *content* of bl is τ, and the *index* of bl is the natural number in $[0, \exp^1(n) - 1]$ (recall that $\exp^1(n) = 2^n$) whose binary code is $bit_1 \ldots bit_n$. The 1-block bl is *initial* (resp., *final*) if $bit_i = 0$ (resp., $bit_i = 1$) for all $1 \leq i \leq n$.

Induction Step: $1 < h \leq k$. An *h-block* is a finite word bl having the form $\$_h \cdot \tau \cdot bl_0 \ldots bl_j \cdot \$_h$ such that $j > 0$, $bl_0, \ldots, bl_j$ are $(h-1)$-blocks, and $\tau \in \{0, 1\}$ if $h < k$, and $\tau \in A \cup (Q \times A)$ otherwise. Additionally, we require that bl_0 is initial, bl_j is final, and for all $0 < i < j$, bl_i is not final. The *content* of bl is τ. The h-block bl is *initial* (resp., *final*) if the content of bl_i is 0 (resp., 1) for all $0 \leq i \leq j$. The h-block bl is *well-formed* if additionally, the following holds: $j = \exp^{h-1}(n) - 1$ and for all $0 \leq i \leq j$, bl_i is well-formed and has index i. If bl is well-formed, then its *index* is the natural number in $[0, \exp^h(n) - 1]$ whose binary code is given by $bit_0, \ldots, bit_j$, where bit_i is the content of the sub-block bl_i for all $0 \leq i \leq j$.

Encoding of (well-formed) TM configurations Let $C = C(0) \ldots C(j)$ be a TM configuration of length at least 2. A *TM configuration code* (for C) is a word over $AP \setminus \{inc, =, good, L, R, \exists, \forall\}$ of the form $code = \tau \cdot bl_0 \ldots bl_j \cdot \tau$ satisfying the following:

- $\tau = \$$ if C is not accepting and $\tau = \$_{acc}$ otherwise;
- for all $0 \leq i \leq j$, bl_i is a k-block whose content is $C(i)$;
- bl_0 is initial and bl_j is the unique final k-block.

We say that *code* is *initial* if C is of the form $(q_\iota, \alpha(0))\,\alpha(1)\ldots\alpha(n)\#\ldots\#$ (note that we do not require that C is the well-formed initial TM configuration). Moreover, we say that *code* is *well-formed* if additionally, $j = \exp^k(n) - 1$ (hence, C is well-formed) and for all $0 \leq i \leq j$, bl_i is well-formed and has index i. Note that there is exactly one well-formed TM configuration code associated with a well-formed TM configuration.

Encoding of TM computations We use the four additional symbols L, R, $\exists$, and $\forall$ to encode single computations of $\mathcal{M}$. Intuitively, the symbol $\exists$ (resp., $\forall$) is used to delimit an existential (resp., universal) configuration, while the symbol L (resp., R) is used to delimit the left (resp., right) successor of a TM configuration. A *TM computation code* is an infinite sequence ν of the form $\nu = \mathit{code}_0 \cdot Q_0 \cdot \mathit{dir}_1 \cdot \mathit{code}_1 \cdot Q_1 \cdot \mathit{dir}_2 \ldots$ such that for all $i \geq 0$, code_i is a TM configuration code which is initial if $i = 0$, $\mathit{dir}_{i+1} \in \{L, R\}$, and $Q_i = \exists$ if the TM configuration C_i associated with code_i is existential, and $Q_i = \forall$ otherwise. The TM computation code ν is well-formed if code_i is well-formed for all $i \geq 0$, and ν is accepting if ν visits some accepting configuration code. Moreover, ν is *fair* if additionally, C_{i+1} is the left successor of C_i if $\mathit{dir}_{i+1} = L$, and the right successor of C_i otherwise. Thus, the accepting, fair, and well-formed TM computation codes encode all the possible accepting TM computations of $\mathcal{M}$ over α.

Note that we have not used the symbols in $\{\mathit{inc}, =, \mathit{good}\}$ in the encoding of TM computations. These extra symbols are used to mark positions along a TM computation code and are crucial for the implementation of transducer satisfying Proposition 30. So, we give the following additional definitions. For a word w over 2^{AP}, the *content* of w is the word over $2^{AP \setminus \{\mathit{inc}, =, \mathit{good}\}}$ obtained by removing from each letter in w the extra symbols in $\{\mathit{inc}, =, \mathit{good}\}$. An *extended TM computation code* is an infinite word over 2^{AP} whose content corresponds to a TM computation code. For an arena with labelling over 2^{AP} and a play π, the *content* of π is the content of its labelling $\mu(\pi)$.

Construction of the arena $\mathcal{G}$ in Proposition 30 The following is straightforward:

Lemma 26. *One can construct in time polynomial in n and the size of $\mathcal{M}$, a 2^{AP}-labelled arena $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$ such that $V_\iota = \{v_\iota\}$ and the following holds:*

1. *for each extended TM computation code ν, there is a play whose labelling is ν;*
2. *for each partial play ρ, the labelling ν_ρ of ρ is the prefix of some extended TM computation code; moreover, for each extended TM computation code ν having ν_ρ as prefix, ρ can be extended to an infinite play whose labelling is ν;*
3. *the set of positions of Player 2 is the set of positions labelled by proposition $\forall$.*

In the following, let $\mathcal{G} = (V, E, V_\iota, v_\iota, \mu)$ be the arena of Lemma 26. Note that in a position labelled by $\forall$, Player 2 chooses between a next position labelled with L and one labelled with R, hence she simulates the universal choices of $\mathcal{M}$; similarly Player 1 simulates the existential choices. Moreover, by Properties 1–3 of Lemma 26, we easily obtain the following.

Remark 19. $\mathcal{M}$ accepts α *iff* Player 1 has a strategy σ in $\mathcal{G}$ such that for all $\pi \in \mathrm{Out}(\sigma)$, the content of $\mu(\pi)$ is a well-formed, fair and accepting TM computation code.

Construction of the finite-state transducer T in Proposition 30 Let ρ be a finite play of $\mathcal{G}$ and $1 \leq h \leq k$. We say that ρ is *not* tagged if each position along ρ is not labelled by the extra symbols in $\{inc, =\}$. Moreover, we say that ρ is a $(h, =)$-*tagged* (resp., (h, inc)-*tagged*) play if exactly two positions along ρ are labelled by some proposition in $\{=, inc\}$, this proposition is $=$ (resp., inc), and these two positions correspond to the initial positions of two h-blocks along ρ. Additionally, for a (h, inc)-tagged play, we require that the two tagged h-blocks are adjacent. Assuming that the two tagged h-blocks bl_1 and bl_2 are well-formed, then the tag $=$ is used to check that bl_1 and bl_2 have the same index, while the tag inc is used to check that the indices of bl_1 and bl_2 are *consecutive* (i.e, bl_1 is not final and the index of bl_2 is the index of bl_1 plus one).

A $\boxminus$-*propositional* $\mathcal{FL}_{\rightsquigarrow}$ *formula* contains only modality $\boxminus$, boolean connectives, and atomic propositions. Let $\mathcal{U}$ be the universe of $\mathcal{G}$. Note that since V_ι is a singleton (Lemma 26), $\mathcal{U}$ is a tree. For a transducer T, a partial play ρ of $\mathcal{G}$, and a $\mathcal{FL}_{\rightsquigarrow}$ state formula φ, we write $\rho \models \varphi$ to mean that $\mathcal{U}, \rho \models \varphi$ with relation $[T]$. Note that if φ is $\boxminus$-propositional, then for each strategy σ of $\mathcal{G}$ and $\rho \in t_\sigma$, $t_\sigma, \rho \models \varphi$ iff $\mathcal{U}, \rho \models \varphi$.

The core result in the proposed reduction is represented by the following lemma.

Lemma 27. *One can construct in time polynomial in n and the size of the TM $\mathcal{M}$ a synchronous finite-state transducer T over 2^{AP} such that there are two* fixed $\boxminus$*-propositional $\mathcal{FL}^{k+1}_{\rightsquigarrow}$ formulas φ_{conf} and φ_{fair} over $\{good\}$, and for all $1 \leq h \leq k$, three* fixed $\boxminus$*-propositional $\mathcal{FL}^{h}_{\rightsquigarrow}$ formulas $\varphi^h_{=}$, φ^h_{inc}, and φ^h_{bl} over $\{good\}$ so that the following holds.*

1. *Let ρ be an $(h, =)$-tagged partial play of $\mathcal{G}$. If the two tagged h-blocks bl_1 and bl_2 of ρ are well-formed, then $\rho \models \varphi^h_{=}$* iff *$bl_1$ and bl_2 have the same index.*
2. *Let ρ be an (h, inc)-tagged partial play of $\mathcal{G}$. If the two tagged h-blocks bl_1 and bl_2 of ρ are well-formed and bl_1 precedes bl_2, then $\rho \models \varphi^h_{inc}$* iff *the indices of bl_1 and bl_2 are consecutive.*
3. *Let ρ be a non-tagged partial play ρ of $\mathcal{G}$ leading to an h-block bl. If each sub-block of bl is well-formed, then $\rho \models \varphi^h_{bl}$* iff *$bl$ is well-formed.*
4. *Let ρ be a non-tagged partial play of ρ leading to a TM configuration code – code – followed by either an $\exists$-position or a $\forall$-position. Then, if each k-block of code is well-formed, $\rho \models \varphi_{conf}$* iff *code is well-formed.*
5. *Let ρ be a non-tagged partial play of $\mathcal{G}$ having a suffix whose content has the form $code \cdot Q \cdot dir \cdot code'$, where $Q \in \{\exists, \forall\}$, $dir \in \{L, R\}$, and code and $code'$ are two TM configuration codes associated with two TM configurations C and C'. If code and $code'$ are well-formed, then $\rho \models \varphi_{fair}$* iff *$C'$ is the dir-successor of C.*

Proof. The main idea, for each point, is to decompose the verification of a property in layers implementable with polynomially many states in the transducer, and invoking other layers thanks to the $\boxminus$ quantifier. Note that for each point of the lemma, the partial plays considered are of a particular form; a transducer can behave differently in each case by first guessing what kind of partial play it will read, behave accordingly and at the same time check that the input indeed is of this form. Thus, for each $i = 1, \ldots, 5$, we describe (Case i, for short) the behavior of the synchronous transducer T on the plays associated with Property i, illustrate the construction of the associated *fixed* $\boxminus$-propositional $\mathcal{FL}_{\rightsquigarrow}$ formula, and at the same time, we prove Property i. Also, when we say that the transducer T reads or write a partial play ρ, we actually refer to its labelling. Case 2 is similar to Case 1, and Case 4 is similar to Case 3. Thus, here, we illustrate only Cases 1, 3, and 5.

Case 1: behavior of T on $(h,=)$-tagged partial plays ρ, construction of $\varphi^h_=$, and proof of Property 1. Let bl_1 and bl_2 be the two tagged h-blocks of ρ. First, assume that $h = 1$. Recall that all 1-blocks are well formed, hence bl_1 and bl_2 are so. On reading the input ρ, T nondeterministically guesses an index $1 \leq j \leq n$ and checks that the j^{th} bit of bl_1 has the same value as the j^{th} bit of bl_2. The output ρ_j generated by T on guessing j has the same content as ρ and its last position is labelled by the proposition *good iff* the check is positive. Thus, by setting $\varphi^1_= := \boxdot\, good$, it holds that bl_1 and bl_2 have the same index *iff* $\rho \models \varphi^1_=$. Hence, Property 1 holds for $h = 1$. Now, assume that $h > 1$. On reading the input ρ, T non-deterministically marks by the symbol $=$, exactly one $(h-1)$-sub-block bl'_1 of bl_1 and one $(h-1)$-sub-block bl'_2 of bl_2, and checks that bl'_1 and bl'_2 have the same content. Thus, the associated output ρ' is an $(h-1,=)$-tagged partial play having the same content as ρ and satisfying the following: the two tagged $(h-1)$-blocks of ρ' are bl'_1 and bl'_2, and the last position of ρ' is labelled by *good iff* bl'_1 and bl'_2 have the same content. Note that this behavior can be implemented by using a number of states polynomial in n and the size of $\mathcal{M}$. Assuming that bl_1 and bl_2 are well-formed, it holds that bl_1 and bl_2 have the same index *iff* for all outputs ρ' generated by T on reading ρ, whenever the two tagged well-formed $(h-1)$-blocks of ρ' have the same index, the last position of ρ' is labelled by *good*. Let $\varphi^h_= := \boxdot(\varphi^{h-1}_= \rightarrow good)$. By the induction hypothesis and assuming that bl_1 and bl_2 are well-formed, it follows that bl_1 and bl_2 have the same index *iff* $\rho \models \varphi^h_=$. Note that by the induction hypothesis, $\varphi^h_=$ is a fixed $\boxdot$-propositional $\mathcal{FL}^h_{\sim}$ formula. Hence, Property 1 holds for $h > 1$ as well.

Behavior of T on (h, inc)-tagged partial plays ρ, construction of φ^h_{inc}, and proof of Property 2. Let bl_1 and bl_2 be the adjacent tagged h-blocks of ρ such that bl_1 precedes bl_2 along ρ. First, assume that $h = 1$. Note that bl_1 and bl_2 are well-formed. Let $1 \leq j \leq n$ be the position of the least significant (*i.e.* rightmost) bit of bl_1 whose value is 0 (note that since bl_1 and bl_2 are adjacent, Lemma 26 and our encoding ensure that such a bit does exists). On reading ρ, T nondeterministically guesses an index $1 \leq i \leq n$ and checks that the following condition is satisfied: if $i < j$, then the i^{th} bit of bl_1 has the same value as the i^{th} bit of bl_2; otherwise the i^{th} bits of bl_1 and bl_2 are distinct. The output ρ_i generated by T on guessing i has the same content as ρ and its last position is labelled by *good iff* the check is positive. Thus, by setting $\varphi^1_{inc} := \boxdot\, good$, it holds that the indices of bl_1 and bl_2 are consecutive *iff* $\rho \models \varphi^1_{inc}$, hence Property 2 holds for $h = 1$. Now, assume that $h > 1$. Let bl_0 be the last $(h-1)$-sub-block of bl_1 whose content is 0 (again, because bl_1 and bl_2 are adjacent, such a block bl_0 does exists). Then, on reading ρ, T non-deterministically marks by the symbol $=$, exactly one $(h-1)$-sub-block bl'_1 of bl_1 and one $(h-1)$-sub-block bl'_2 of bl_2, and checks that the following condition is satisfied: (*) if bl'_1 precedes bl_0, then bl'_1 and bl'_2 have the same content; otherwise, bl'_1 and bl'_2 have distinct contents. Thus, the associated output ρ' is an $(h-1,=)$-tagged partial play having the same content as ρ and satisfying the following: the two tagged $(h-1)$-blocks of ρ' are bl'_1 and bl'_2, and the last position of ρ' is labelled by *good iff* the above condition (*) is satisfied. Note that this behavior can be implemented by using a number of states polynomial in n and the size of $\mathcal{M}$. Thus, assuming that bl_1 and bl_2 are well-formed, it holds that the indices of bl_1 and bl_2 are consecutive *iff* for all outputs ρ' generated by T on reading ρ, whenever the two tagged well-formed $(h-1)$-blocks of ρ' have the same index, the last position of ρ' is labelled by *good*. Let $\varphi^h_{inc} := \boxdot(\varphi^{h-1}_= \rightarrow good)$. Note that by the proof of Property 1,

φ^h_{inc} is a fixed $\boxdot$-propositional $\mathcal{FL}^h_{\rightsquigarrow}$ formula. By Property 1 and assuming that bl_1 and bl_2 are well-formed, it follows that the indices of bl_1 and bl_2 are consecutive *iff* $\rho \models \varphi^h_{inc}$. Hence, Property 2 holds for $h > 1$ as well.

Case 3: behavior of T on non-tagged partial plays ρ leading to h-blocks bl, construction of φ^h_{bl}, and proof of Property 3. First, assume that $h = 1$. On reading ρ, T simply outputs ρ itself. Since a 1-block is always well-formed, by setting $\varphi^1_{bl} := \texttt{true}$, Property 3 holds for $h = 1$. Now, assume that $h > 1$. On reading the input ρ which leads to an h-block bl, T non-deterministically marks with the symbol *inc* exactly two adjacent $(h-1)$-sub-blocks bl_1 and bl_2 of bl, and checks that the last $(h-1)$-sub-block is final. Thus, the associated output ρ' is an $(h-1, inc)$-tagged partial play having the same content as ρ and satisfying the following: the two tagged $(h-1)$-blocks of ρ' are bl_1 and bl_2, and the last position of ρ' is labelled by *good iff* the last $(h-1)$-sub-block of bl is final. Thus, assuming that all the sub-blocks of bl are well-formed, it holds that bl is well-formed *iff* for all outputs ρ' generated by T on reading ρ, the indices of the two tagged well-formed $(h-1)$-blocks of ρ' are consecutive and the last position of ρ' is labelled by *good*. Let $\varphi^h_{bl} := \boxdot(\varphi^{h-1}_{inc} \wedge good)$. Note that by Property 2, φ^h_{bl} is a fixed $\boxdot$-propositional $\mathcal{FL}^h_{\rightsquigarrow}$ formula. Then, by Property 2 and assuming that all the sub-blocks of bl are well-formed, it follows that bl is well-formed *iff* $\rho \models \varphi^h_{bl}$. Hence, Property 3 holds for $h > 1$ as well.

Behavior of T on the non-tagged partial plays ρ of Property 4, construction of φ_{conf}, and proof of Property 4. The behavior of T on this kind of plays is similar to the behavior of T on non-tagged plays leading to h-blocks, and the proof of Property 4 is similar to the proof of Property 3. In particular, $\varphi_{conf} := \boxdot\, \varphi^k_{inc}$.

Case 5: behavior of T on the non-tagged partial plays ρ of Property 5, construction of φ_{fair}, and proof of Property 5. First, we need an additional notation. Let $C = u_0 \dots u_{\exp^k(n)-1}$ be a well-formed TM configuration and $dir = L$ (resp., $dir = R$). For all $0 \leq i \leq \exp^k(n) - 1$, the value u'_i of the i^{th} cell of the left (resp., right) $\mathcal{M}$-successor of C is completely determined by the values u_{i-1}, u_i and u_{i+1} (taking u_{i+1} for $i = \exp^h(n) - 1$ and u_{i-1} for $i = 0$ to be some special symbol, say '$\perp$'). Let $next_{dir}(u_{i-1}, u_i, u_{i+1})$ be our expectation for u'_i (this function can be trivially obtained from the transition function δ of $\mathcal{M}$). Let ρ be a non-tagged partial play of $\mathcal{G}$ having a suffix whose content has the form $code \cdot Q \cdot dir \cdot code'$, where $Q \in \{\exists, \forall\}$, $dir \in \{L, R\}$, and $code$ and $code'$ are two TM configuration codes associated with two TM configurations C and C'. Assume that $dir = L$ (the other case is similar). The behavior of T on the input ρ is as follows. T non-deterministically marks with the symbol '=' exactly one k-block bl of $code$ and one k-block bl' of $code'$, and checks that $u' = next_L(u_-, u, u_+)$, where u (resp., u') is the content of bl (resp., bl') and u_- (resp., u_+) is the content of the k-block that precedes (resp., follows) bl in $code$ if it exists, and the special symbol $\perp$ otherwise. Thus, the associated output ρ' is a $(k, =)$-tagged partial play having the same content as ρ and satisfying the following: the two tagged k-blocks of ρ' are bl and bl', and the last position of ρ' is labelled by *good iff* $u' = next_L(u_-, u, u_+)$. Note that this behavior can be implemented by using a number of states polynomial in n and the size of $\mathcal{M}$. Thus, assuming that $code$ and $code'$ are well-formed, it holds that C' is the left $\mathcal{M}$-successor of C *iff* for all outputs ρ' generated by T on reading ρ, whenever the two tagged well-formed k-blocks of ρ' have the same index, the last position of ρ' is labelled by *good*. Let $\varphi_{fair} := \boxdot(\varphi^k_{=} \rightarrow good)$. Note that

φ_{fair} is a fixed $\boxdot$-propositional $\mathcal{FL}^{k+1}_{\rightsquigarrow}$ formula. By Property 1 and assuming that *code* and *code*$'$ are well-formed, it follows that C' is the left $\mathcal{M}$-successor of C *iff* $\rho \models \varphi_{fair}$. Hence, Property 5 holds, which concludes the proof of the proposition. $\square$

Construction of the *fixed* $\mathcal{FL}^{k+1}_{\rightsquigarrow}$ formula φ in Proposition 30 and proof of Proposition 30 Let T be the synchronous finite-state transducer and φ_{conf}, φ_{fair}, φ^1_{bl}, ..., φ^k_{bl} be the *fixed* $\boxdot$-propositional $\mathcal{FL}^{k+1}_{\rightsquigarrow}$ formulas satisfying Lemma 27. Then, the *fixed* $\mathcal{FL}^{k+1}_{\rightsquigarrow}$ formula φ is given by $\varphi := \mathbf{A}\psi$, the path formula ψ being defined as follows, where $\varphi' := \$_1 \vee \ldots \vee \$_k \vee \$ \vee \$_{acc}$:

$$\psi := \begin{array}{c} \underbrace{\mathbf{G}\big(\neg inc \wedge \neg =\big) \wedge \mathbf{GF}(\$ \vee \$_{acc})}_{\text{the infinite play corresponds to a non-tagged TM computation code}} \\ \wedge \\ \underbrace{\bigwedge_{h=1}^{k} \mathbf{G}((\$_h \wedge \mathbf{X}\varphi') \rightarrow \varphi^h_{bl}) \wedge \mathbf{G}((\exists \vee \forall) \rightarrow \varphi_{conf})}_{\text{the TM computation code is well-formed}} \\ \wedge \\ \underbrace{\mathbf{G}\big((\exists \vee \forall) \rightarrow \mathbf{XG}(\mathbf{X}(\exists \vee \forall) \rightarrow \varphi_{fair})\big)}_{\text{the well-formed TM sequence code is fair}} \\ \wedge \\ \underbrace{\mathbf{F}\,\$_{acc}}_{\text{the fair well-formed TM computation code is accepting}} \end{array}$$

By Remark 19 and Lemma 27, it easily follows that $\mathcal{M}$ accepts α *iff* there is a strategy σ of Player 1 in $\mathcal{G}$ such that the contents of all the plays in the outcome of σ are accepting, fair, and well-formed TM computation codes *iff* there is a strategy σ of Player 1 in $\mathcal{G}$ such that for the relation $[T]$, $t_\sigma \models \mathbf{A}\psi$ *iff* Player 1 has a $([T], \mathbf{A}\psi)$-uniform strategy in $\mathcal{G}$, which proves Proposition 30.

Bibliography

de Alfaro L. and Henzinger T.A. (2000), Concurrent omega-regular games. in *LICS*, pp. 141–154.

de Alfaro L., Henzinger T.A. and Kupferman O. (1998), Concurrent reachability games. in *FOCS*, pp. 564–575 (IEEE Computer Society).

Alur R., Henzinger T.A. and Kupferman O. (2002), Alternating-time temporal logic. *J. ACM*, vol. 49(5): pp. 672–713.

Aucher G. (2013), Infinite Games in Epistemic Temporal Logic via Supervisory Control Theory. Rapport de recherche RR-8374, INRIA. URL `http://hal.inria.fr/hal-00866155`.

Aucher G. and Bolander T. (2013), Undecidability in epistemic planning. in *IJCAI*.

Aucher G. and Herzig A. (2011), Exploring the power of converse events. in *Dynamic formal epistemology*, pp. 51–74 (Springer), doi:10.1007/978-94-007-0074-1_4.

Aucher G., Maubert B. and Schwarzentruber F. (2011), Tableau Method and NEXPTIME-Completeness of DEL-Sequents. in *Methods for Modalities (M4M)*, vol. 278, pp. 17–30 (Electr. Notes Theor. Comput. Sci.).

Aucher G., Maubert B. and Schwarzentruber F. (2012), Generalized DEL-Sequents. in *JELIA*, vol. 7519 of *Lecture Notes in Computer Science*, edited by L.F. del Cerro, A. Herzig and J. Mengin, pp. 54–66 (Springer), ISBN 978-3-642-33352-1.

Baltag A., Moss L.S. and Solecki S. (1998), The logic of public announcements, common knowledge, and private suspicions. in *Proceedings of the 7th conference on Theoretical aspects of rationality and knowledge*, pp. 43–56 (Morgan Kaufmann Publishers Inc.).

Bekker W. and Goranko V. (2007), Symbolic model checking of tense logics on rational kripke models. in *ILC*, vol. 5489 of *Lecture Notes in Computer Science*, edited by M. Archibald, V. Brattka, V. Goranko and B. Löwe, pp. 2–20 (Springer), ISBN 978-3-642-03091-8.

van Benthem J. (2001), Games in Dynamic-Epistemic Logic. *Bulletin of Economic Research*, vol. 53(4): pp. 219–248, doi:10.1111/1467-8586.00133.

van Benthem J. (2005), The Epistemic Logic of IF Games. *The Philosophy of Jaakko Hintikka*, vol. 30.

van Benthem J. (2011), *Logical dynamics of information and interaction* (Cambridge University Press).

van Benthem J., Gerbrandy J., Hoshi T. and Pacuit E. (2009), Merging frameworks for interaction. *Journal of Philosophical Logic*, vol. 38(5): pp. 491–526, doi:10.1007/s10992-008-9099-x.

van Benthem J. and Liu F. (2004), Diversity of logical agents in games. *Philosophia Scientiae*, vol. 8(2): pp. 163–178.

Bernet J., Janin D. and Walukiewicz I. (2002), Permissive strategies: from parity games to safety games. *ITA*, vol. 36(3): pp. 261–275.

Berstel J. (1979), *Transductions and context-free languages*, vol. 4 (Teubner Stuttgart), doi:10.1007/978-3-663-09367-1.

Berwanger D., Chatterjee K., Wulf M.D., Doyen L. and Henzinger T.A. (2010), Strategy construction for parity games with imperfect information. *Inf. Comput.*, vol. 208(10): pp. 1206–1220.

Berwanger D. and Doyen L. (2008), On the power of imperfect information. in *FSTTCS*, vol. 2 of *LIPIcs*, edited by R. Hariharan, M. Mukund and V. Vinay, pp. 73–82 (Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik).

Berwanger D. and Kaiser L. (2010), Information tracking in games on graphs. *Journal of Logic, Language and Information*, vol. 19(4): pp. 395–412.

Berwanger D., Kaiser L. and Lessenich S. (2012), Solving counter parity games. in *MFCS*, vol. 7464 of *Lecture Notes in Computer Science*, edited by B. Rovan, V. Sassone and P. Widmayer, pp. 160–171 (Springer), ISBN 978-3-642-32588-5.

Blackburn P., van Benthem J.F. and Wolter F. (2006), *Handbook of modal logic*, vol. 3 (Elsevier).

Blumensath A. and Grädel E. (2000), Automatic structures. in *LICS*, pp. 51–62 (IEEE Computer Society), ISBN 0-7695-0725-5.

Blumensath A. and Grädel E. (2004), Finite presentations of infinite structures: Automata and interpretations. *Theory Comput. Syst.*, vol. 37(6): pp. 641–674.

Bojanczyk M. (2002), Two-way alternating automata and finite models. in *ICALP*, vol. 2380 of *Lecture Notes in Computer Science*, edited by P. Widmayer, F.T. Ruiz, R.M. Bueno, M. Hennessy, S. Eidenbenz and R. Conejo, pp. 833–844 (Springer), ISBN 3-540-43864-5.

Bolander T. and Andersen M.B. (2011), Epistemic planning for single and multi-agent systems. *Journal of Applied Non-Classical Logics*, vol. 21(1): pp. 9–34.

Chandra A.K., Kozen D. and Stockmeyer L.J. (1981), Alternation. *Journal of the ACM*, vol. 28(1): pp. 114–133.

Chatterjee K., Doyen L., Henzinger T.A. and Raskin J.F. (2006), Algorithms for omega-regular games with imperfect information' . in *CSL*, vol. 4207 of *Lecture Notes in Computer Science*, edited by Z. Ésik, pp. 287–302 (Springer), ISBN 3-540-45458-6.

Clarke E.M. and Emerson E.A. (1981), Design and synthesis of synchronization skeletons using branching-time temporal logic. in *Logic of Programs*, vol. 131 of *Lecture Notes in Computer Science*, edited by D. Kozen, pp. 52–71 (Springer), ISBN 3-540-11212-X.

Dégremont C., Löwe B. and Witzel A. (2011), The synchronicity of dynamic epistemic logic. in *TARK*, edited by K.R. Apt, pp. 145–152 (ACM), ISBN 978-1-4503-0707-9.

Dima C. (2008), Revisiting satisfiability and model-checking for CTLK with synchrony and perfect recall. in *CLIMA*, vol. 5405 of *Lecture Notes in Computer Science*, edited by M. Fisher, F. Sadri and M. Thielscher, pp. 117–131 (Springer), ISBN 978-3-642-02733-8.

Dima C., Enea C. and Guelev D.P. (2010), Model-checking an alternating-time temporal logic with knowledge, imperfect information, perfect recall and communicating coalitions. *Electronic Proceedings in Theoretical Computer Science*, vol. 25, doi:10.4204.

van Ditmarsch H. and Kooi B. (2006), Semantic results for ontic and epistemic change. *Logic and the Foundations of Game and Decision Theory (LOFT 7)*, p. 87.

van Ditmarsch H.P., Van der Hoek W. and Kooi B.P. (2007), *Dynamic epistemic logic*, vol. 337 (Springer).

Dziembowski S., Jurdzinski M. and Walukiewicz I. (1997), How much memory is needed to win infinite games? in *LICS*, pp. 99–110 (IEEE), doi:10.1109/LICS.1997.614939.

van Eijck J. (2004), Guarded actions. Tech. Rep., Technical Report SEN-E0425, CWI, Amsterdam.

Eilenberg S. (1974), *Automata, languages, and machines*, vol. 1 (Elsevier).

Emerson E.A. (1990), Temporal and modal logic. in *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pp. 995–1072 (MIT Press).

Emerson E.A. and Halpern J.Y. (1983), "Sometimes" and "Not Never" revisited: On branching versus linear time. in *POPL*, edited by J.R. Wright, L. Landweber, A.J. Demers and T. Teitelbaum, pp. 127–140 (ACM Press), ISBN 0-89791-090-7.

Emerson E.A. and Lei C.L. (1986), Efficient model checking in fragments of the propositional mu-calculus (extended abstract). in *LICS*, pp. 267–278 (IEEE Computer Society).

Engström F. (2012), Generalized quantifiers in dependence logic. *Journal of Logic, Language and Information*, vol. 21(3): pp. 299–324.

Fagin R., Halpern J. and Vardi M. (1991), A model-theoretic analysis of knowledge. *Journal of the ACM (JACM)*, vol. 38(2): pp. 382–428, doi:10.1145/103516.128680.

Fagin R., Halpern J.Y., Moses Y. and Vardi M.Y. (1995), *Reasoning about knowledge*, vol. 4 (MIT press Cambridge).

Frougny C. and Sakarovitch J. (1993), Synchronized rational relations of finite and infinite words. *Theor. Comput. Sci.*, vol. 108(1): pp. 45–82.

Galliani P. (2012), Inclusion and exclusion dependencies in team semantics - on some logics of imperfect information. *Ann. Pure Appl. Logic*, vol. 163(1): pp. 68–84.

Ghallab M., Nau D.S. and Traverso P. (2004), *Automated planning - theory and practice* (Elsevier), ISBN 978-1-55860-856-6.

Grädel E. (2013), Model-checking games for logics of imperfect information. *Theoretical Computer Science*, vol. 493: pp. 2–14.

Grädel E., Thomas W. and Wilke T. (editors) (2002), *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, vol. 2500 of *Lecture Notes in Computer Science* (Springer), ISBN 3-540-00388-6.

Grädel E. and Väänänen J.A. (2013), Dependence and independence. *Studia Logica*, vol. 101(2): pp. 399–410.

Halpern J.Y., van der Meyden R. and Vardi M.Y. (2004), Complete Axiomatizations for Reasoning about Knowledge and Time. *SIAM J. Comput.*, vol. 33(3): pp. 674–703.

Halpern J.Y. and Vardi M.Y. (1989), The complexity of reasoning about knowledge and time. 1. Lower bounds. *Journal of Computer and System Sciences*, vol. 38(1): pp. 195–237, doi:10.1145/12130.12161.

Henkin L. (1961), Some remarks on infinitely long formulas. in *Infinitistic Methods, Warsaw*, pp. 167–183.

Hintikka J. (1962), *Knowledge and belief*, vol. 13 (Cornell University Press Ithaca).

Hintikka J. and Sandu G. (1989), Informational independence as a semantical phenomenon. *Studies in Logic and the Foundations of Mathematics*, vol. 126: pp. 571–589.

Hodges W. (1997), Compositional semantics for a language of imperfect information. *Logic Journal of the IGPL*, vol. 5(4): pp. 539–563.

van der Hoek W. and Wooldridge M. (2003), Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, vol. 75(1): pp. 125–157, doi:10.1023/A:1026185103185.

Hopcroft J.E., Motwani R. and Ullman J.D. (2003), *Introduction to automata theory, languages, and computation - international edition (2. ed)* (Addison-Wesley), ISBN 978-0-321-21029-6.

Hoshi T. and Yap A. (2009), Dynamic epistemic logic with branching temporal structures. *Synthese*, vol. 169(2): pp. 259–281.

Hossley R. and Rackoff C. (1972), The emptiness problem for automata on infinite trees. in *13th Annual Symposium on Switching and Automata Theory*, pp. 121–124 (IEEE).

Jamroga W. and van der Hoek W. (2004), Agents that know how to play. *Fundam. Inform.*, vol. 63(2-3): pp. 185–219.

Johnson J.H. (1986), Rational equivalence relations. *Theor. Comput. Sci.*, vol. 47(3): pp. 39–60.

Kanellakis P.C. and Smolka S.A. (1990), Ccs expressions, finite state processes, and three problems of equivalence. *Information and Computation*, vol. 86(1): pp. 43–68.

Khoussainov B. and Nerode A. (1994), Automatic presentations of structures. in *LCC*, vol. 960 of *Lecture Notes in Computer Science*, edited by D. Leivant, pp. 367–392 (Springer), ISBN 3-540-60178-3.

Khoussainov B., Nies A., Rubin S. and Stephan F. (2007), Automatic structures: Richness and limitations. *Logical Methods in Computer Science*, vol. 3(2).

Kozen D. (1983), Results on the propositional mu-calculus. *Theor. Comput. Sci.*, vol. 27: pp. 333–354.

Kupferman O. and Vardi M.Y. (1997), Module checking revisited. in *CAV*, vol. 1254 of *Lecture Notes in Computer Science*, edited by O. Grumberg, pp. 36–47 (Springer), ISBN 3-540-63166-6.

Kupferman O., Vardi M.Y. and Wolper P. (2000), An automata-theoretic approach to branching-time model checking. *J. ACM*, vol. 47(2): pp. 312–360.

Kupferman O., Vardi M.Y. and Wolper P. (2001), Module checking. *Inf. Comput.*, vol. 164(2): pp. 322–344.

Ladner R.E. and Reif J.H. (1986), The Logic of Distributed Protocols. in *TARK*, edited by J.Y. Halpern, pp. 207–222 (Morgan Kaufmann), ISBN 0-934613-04-4.

Lehmann D. (1984), Knowledge, common knowledge and related puzzles (extended summary). in *Proceedings of the third annual ACM symposium on Principles of distributed computing*, pp. 62–67 (ACM), doi:10.1145/800222.806736.

Liu F. (2009), Diversity of agents and their interaction. *Journal of Logic, Language and Information*, vol. 18(1): pp. 23–53.

Löding C. (2014), Automata on infinite trees. in *preliminary version for the handbook Automata: from Mathematics to Applications*, edited by J.E. Pin. To appear.

Löwe B., Pacuit E. and Witzel A. (2011), DEL planning and some tractable cases. in *LORI*, vol. 6953 of *Lecture Notes in Computer Science*, edited by H.P. van Ditmarsch, J. Lang and S. Ju, pp. 179–192 (Springer), ISBN 978-3-642-24129-1.

Maubert B. and Pinchinat S. (2009), Games with opacity condition. in *Reachability Problems*, vol. 5797 of *Lecture Notes in Computer Science*, edited by O. Bournez and I. Potapov, pp. 166–175 (Springer), ISBN 978-3-642-04419-9.

Maubert B. and Pinchinat S. (2012), Uniform strategies. in *LOFT 2012, 10th Conference on Logic and the Foundations of Game and Decision Theory, Sevilla, Spain*. URL `http://www.irisa.fr/prive/bmaubert/publications.html`.

Maubert B. and Pinchinat S. (2013), Jumping automata for uniform strategies. in *Proceedings of the 33rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'13)*, edited by A. Seth and N. Vishnoi, Leibniz International Proceedings in Informatics, pp. 287–298 (Leibniz-Zentrum für Informatik, Guwahati, India), doi:10.4230/LIPIcs.FSTTCS.2013.287.

Maubert B. and Pinchinat S. (2014), A general notion of uniform strategies. *International Game Theory Review*, vol. 16(01), doi:10.1142/S0219198914400040.

Maubert B., Pinchinat S. and Bozzelli L. (2011), Opacity issues in games with imperfect information. in *GandALF*, vol. 54 of *EPTCS*, edited by G. D'Agostino and S.L. Torre, pp. 87–101, doi:10.4204/EPTCS.54.7.

Maubert B., Pinchinat S. and Bozzelli L. (2013), The complexity of synthesizing uniform strategies. in *Proceedings 1st International Workshop on Strategic Reasoning*, vol. 112 of *EPTCS*, edited by F. Mogavero, A. Murano and M.Y. Vardi, pp. 115–122.

McNaughton R. (1993), Infinite games played on finite graphs. *Ann. Pure Appl. Logic*, vol. 65(2): pp. 149–184.

Meduna A. and Zemek P. (2012), Jumping finite automata. *International Journal of Foundations of Computer Science*, vol. 23(07): pp. 1555–1578.

van der Meyden R. and Shilov N.V. (1999), Model checking knowledge and time in systems with perfect recall (extended abstract). in *FSTTCS*, vol. 1738 of *Lecture Notes in Computer Science*, edited by C.P. Rangan, V. Raman and R. Ramanujam, pp. 432–445 (Springer), ISBN 3-540-66836-5.

van der Meyden R. and Vardi M.Y. (1998), Synthesis from knowledge-based specifications. in *CONCUR'98 Concurrency Theory*, pp. 34–49 (Springer).

van der Meyden R. and Wilke T. (2005), Synthesis of distributed systems from knowledge-based specifications. in *CONCUR*, vol. 3653 of *Lecture Notes in Computer Science*, edited by M. Abadi and L. de Alfaro, pp. 562–576 (Springer), ISBN 3-540-28309-9.

Morvan C. (2000), On rational graphs. in *FoSSaCS*, vol. 1784 of *Lecture Notes in Computer Science*, edited by J. Tiuryn, pp. 252–266 (Springer), ISBN 3-540-67257-5.

Morvan C. and Rispal C. (2005), Families of automata characterizing context-sensitive languages. *Acta Inf.*, vol. 41(4-5): pp. 293–314.

Muller D.E. and Schupp P.E. (1987), Alternating automata on infinite trees. *Theoretical computer science*, vol. 54(2): pp. 267–276.

Muller D.E. and Schupp P.E. (1995), Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of rabin, mcnaughton and safra. *Theor. Comput. Sci.*, vol. 141(1&2): pp. 69–107.

Pacuit E. (2007), Some comments on history based structures. *Journal of Applied Logic*, vol. 5(4): pp. 613–624.

Pacuit E. and van Benthem J. (2006), The tree of knowledge in action: Towards a common perspective. *Proceedings of Advances in Modal Logic Volume 6*, pp. 87–106.

Parikh R. and Ramanujam R. (1985), Distributed processes and the logic of knowledge. in *Logic of Programs*, vol. 193 of *Lecture Notes in Computer Science*, edited by R. Parikh, pp. 256–268 (Springer), ISBN 3-540-15648-8.

Parikh R. and Ramanujam R. (2003), A knowledge based semantics of messages. *Journal of Logic, Language and Information*, vol. 12(4): pp. 453–467.

Peterson G., Reif J.H. and Azhar S. (2001), Lower bounds for multiplayer noncooperative games of incomplete information. *Computers & Mathematics with Applications*, vol. 41(7): pp. 957–992.

Piccione M. and Rubinstein A. (1997), The absent-minded driver's paradox: synthesis and responses. *Games and Economic Behavior*, vol. 20(1): pp. 121–130.

Piterman N. and Vardi M.Y. (2004), Global model-checking of infinite-state systems. in *CAV*, vol. 3114 of *Lecture Notes in Computer Science*, edited by R. Alur and D. Peled, pp. 387–400 (Springer), ISBN 3-540-22342-8.

Pnueli A. (1977), The temporal logic of programs. in *FOCS*, pp. 46–57 (IEEE Computer Society).

Pnueli A. and Rosner R. (1989), On the synthesis of an asynchronous reactive module. in *Proc. 16th Int. Coll. on Automata, Languages and Programming, ICALP'89, Stresa, Italy, LNCS 372*, pp. 652–671 (Springer-Verlag), doi:10.1007/BFb0035790.

Puchala B. (2010), Asynchronous omega-regular games with partial information. in *MFCS*, vol. 6281 of *Lecture Notes in Computer Science*, edited by P. Hlinený and A. Kucera, pp. 592–603 (Springer), ISBN 978-3-642-15154-5.

Ramadge P.J. and Wonham W.M. (1987), Supervisory control of a class of discrete event processes. *SIAM journal on control and optimization*, vol. 25(1): pp. 206–230.

Reif J.H. (1984), The complexity of two-player games of incomplete information. *Journal of computer and system sciences*, vol. 29(2): pp. 274–301, ISSN 0022-0000, doi:10.1016/0022-0000(84)90034-5.

Riedweg S. and Pinchinat S. (2003), Quantified mu-calculus for control synthesis. in *MFCS*, vol. 2747 of *Lecture Notes in Computer Science*, edited by B. Rovan and P. Vojtás, pp. 642–651 (Springer), ISBN 3-540-40671-9.

Sato J. (1977), A study of kripke style methods for some modal logic by gentzen's sequential method. Tech. Rep., Technical report, Publication Research Institute for Mathematical Science.

Thomas W. (1990), Automata on infinite objects. in *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pp. 133–192 (Elsevier).

Väänänen J.A. (2007), *Dependence Logic - A New Approach to Independence Friendly Logic*, vol. 70 of *London Mathematical Society student texts* (Cambridge University Press), ISBN 978-0-521-70015-3.

Vardi M.Y. (1995), Alternating automata and program verification. in *Computer Science Today*, vol. 1000 of *Lecture Notes in Computer Science*, edited by J. van Leeuwen, pp. 471–485 (Springer), ISBN 3-540-60105-8.

Vardi M.Y. (1998), Reasoning about the past with two-way automata. in *ICALP*, vol. 1443 of *Lecture Notes in Computer Science*, edited by K.G. Larsen, S. Skyum and G. Winskel, pp. 628–641 (Springer), ISBN 3-540-64781-3.

Vardi M.Y. (2008), From Church and Prior to PSL. in *25 Years of Model Checking*, vol. 5000 of *Lecture Notes in Computer Science*, edited by O. Grumberg and H. Veith, pp. 150–171 (Springer), ISBN 978-3-540-69849-4.

Vardi M.Y. and Wolper P. (1994), Reasoning about infinite computations. *Information and Computation*, vol. 115(1): pp. 1–37.

Wang Y. and Aucher G. (2013), An alternative axiomatization of DEL and its applications. in *IJCAI*.

Yu Q., Wen X. and Liu Y. (2013), Multi-agent epistemic explanatory diagnosis via reasoning about actions. in *IJCAI*, pp. 1183–1190.

Zielonka W. (1998), Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, vol. 200(1-2): pp. 135–183.

Index

Résumé en français

On trouve dans la littérature de nombreux exemples de jeux où les stratégies souhaitées sont soumises à des contraintes "transversales" portant sur des ensembles de parties, reliées entre elles par quelque relation sémantique. L'exemple le plus fameux est celui des stratégies dans les jeux à information imparfaite, et les jeux où la condition de gain a un aspect épistémique en sont d'autres. Cependant, aucune étude approfondie n'a à notre connaissance été menée sur ce type de contraintes dans leur généralité. C'est ce que nous nous proposons de commencer dans cette thèse.

Nous définissons donc une notion générale de stratégies uniformes. Les propriétés d'uniformité des stratégies sont exprimées dans un langage logique qui étend CTL^* avec deux quantificateurs originaux. Ces quantificateurs sont très proches des opérateurs de connaissance classiques en logique épistémique, et font intervenir des ensembles de parties reliées entre elles par des relations binaires. Nous montrons comment cette notion de stratégies uniformes capture les exemples connus de la littérature, puis nous étudions en profondeur le problème de la synthèse de stratégies uniformes, en considérant que les relations binaires entre les parties sont reconnaissables par des automates finis (relations rationnelles). Nous établissons plusieurs résultats de décidabilité et de complexité, reposant largement sur des techniques d'automates : nous introduisons notamment comme outils les automates d'arbres bondissants et les automates d'ensembles d'informations. Par ailleurs, nos résultats permettent d'améliorer des résultats existants et d'en établir de nouveaux, dans les domaines du model-checking des logiques temporelles et épistémiques, ainsi que de la planification épistémique.

Summary in English

There are in the literature many examples of games where the desired strategies are submitted to "transversal" constraints involving sets of plays, related by some semantic relation. The most famous example is strategies for games with imperfect information, and games where the objective involves some epistemic aspect provide many more examples. Nevertheless, to the best of our knowledge, there has been no thorough study on this type of constraints in their generality. This is what this thesis intends to start.

Therefore, we define a general notion of uniform strategies. Uniformity properties of strategies are expressed in a logical language that extends CTL^* with two original quantifiers. These quantifiers are very close to the classic knowledge operators of epistemic logics, and they involve sets of plays related by binary relations. We show how this notion of uniform strategies captures the known examples from the literature, and we study in depth the problem of uniform strategy synthesis, assuming that the binary relations between plays can be recognized by finite automata (rational relations). We establish several decidability and complexity results, relying widely on automata techniques: in particular, we introduce as tools jumping tree automata and information sets automata. Moreover, our results enable us to improve existing results and establish new ones, in the domains of model checking epistemic temporal logics, and epistemic planning.